# Title: Action Classification

**Project Team Members:**

1. Sri Rupin Potula
2. Vibha Swaminathan
3. Vibhu Dixit
4. Abhishek Mulasi

## Abstract:

The main objective of Phase-2 is to extract the features from images. To do this, the hmdb51 dataset has been taken for the study. This dataset provided has about 51 actions. But for the study, different action types are taken namely golf, shoot, ball, brush, hair, handstand, shoot, bow, cartwheel, hit, shoot, gun, cartwheel, hug, sit, catch, jump, situp, chew, kick, smile, clap, kick, ball, smoke, climb, somersault, climb, stairs, laugh and stand. These are split into target and non-target video directories. To obtain the features effectively, a pre-trained model namely r3d_18 is used to gather the output at specific layers. Then, for the videos depending on the STIP (interest points) HoG and HoF are constructed. For better visualization of the video frames a color histogram is featured to gather the different spatial orientations of different parts of the same size. Dimensionality reduction techniques are used to extract features better using latent semantics. The features are PCA, SVD, LDA, and KMeans. As a result, the feature descriptors are constructed for all the videos. As a result, for any given video the nearest videos are gathered along with similarity scores. Even predicted the class label for the target videos.

**Keywords:** r3d_18, OpenCV, tensor, clustering, manhattan distance, Euclidean distance, histogram, STIP, HoG, HoF, Cosine Similarity, PCA, SVD, KMeans, LDA.

## 1. Assumption:

Through our implementation, Color Histogram representation was not considered as integrating color histogram into the features would break the existing code. For Task-3 and Task-4, for all dimensionality reduction techniques, we ask the user for the feature space used in Task-2. For Task-4, through this implementation, target video labels are considered as these are stored as part of Task-0a, for non-target videos the labels to video mappings weren't present. For Task-5 and Task-6, which were assigned to Vibha Swaminathan, the code was not produced based on the requirements provided. The code uses pre-existing libraries for PCA and SVD. The wrong LDA technique (Linear discriminant analysis instead of Latent Dirichlet Allocation) was used in these 2 tasks. The code was not integrated with the rest of the source code and initializes the features with random values.

## 2. Introduction:

In the digital era, the number of videos that are being captured has increased rapidly because of the increase in the use of mobiles, laptops, and tablets. With that rapid increase, the volume of videos has been increasing exponentially. So, it is important to understand the content of the video structure, specifically the action that was being performed in the video. The project focuses on advanced techniques for video analysis. We analyze different approaches to vector modeling, similarity measurement, and feature extraction in video data. The primary goals consist of:

1. Using the R3D-18 architecture to implement feature descriptors based on neural networks.
2. Using HoG and HoF descriptors, develop a Spatio-Temporal Interest Point (STIP) analysis.
3. Converting video frames into color histograms.
4. Taking a set of target videos' feature descriptors and storing them.
5. Putting in place a system for comparing similar videos to recognize and display them.

We use libraries like PyTorch, torchvision, PyAV, and OpenCV for deep learning and visual information extraction. With potential applications in duplicate detection and video recommendation systems, the project aims to improve the field of content-based video access and evaluation.

## 3. Correction of Phase-1:

### 3.1 Task - 1:

In Phase 1's Task 1, the output generated for a given query video was not up to the mark. The extraction of the features is corrected. Originally, videos with less than 32 frames were skipped. As this would cause us to skip many videos in Phase 2, this has been fixed by appending the last frame.

### 3.2 Task - 2:

The video was not visualized, and the output BOF-HOG and BOF-HOF were only displayed as texts. These issues have been fixed using OpenCV to visualize the video and plot the BOF-HOG and BOF-HOF vectors as 12 adjacent histograms.

### 3.3 Task - 3:

The output format was not as per the requirements, this has been corrected to display the appropriate histogram vectors. The vectors have also been normalized correctly. Fixed the issue where some videos have problems with the encoding (causing miscalculations) resulting in the files being processed faster than before. The bins are mapped by getting the color clusters using the k-means model. The model is trained based on 5 action video folders.

## 4. Choice of Database:

Initially, ‘JSON’ is used to store all the features that are specific features to a video including the videoID to video mapping. JSON is used as the video and its respective features can be retrieved easily using indexing. Along with this, SQLite database is also used as it is an embedded database and is not server-sided. Query searching is very straightforward for anyone without any specific knowledge about the source code. SQLite is highly capable of handling a database such as ours in terms of size.[1]

## 5. Description of the Proposed Solution:

### 5.1 Task - 0:

The entire dataset as well as the STIP files are downloaded from the source website.[2] The .rar files are unrarred using tools like WinRar or online unrarring tools.[3]

#### 5.1.1 Task - 0a:

As part of this task, a list was made that contains all the target video action labels. Two separate directories for target videos and non-target videos are made in the dataset directory and all the videos are segregated under these directories. The function ‘move_stips()’ moves the video stip files into dataset_stips/target_videos or dataset_stips/non_target_videos. The function, ‘create_video_id’ was used to create the videoID for all the videos starting from 0. For every video, a unique ID was assigned and the values were stored in a JSON named ‘VideoID.json’. Using the generated videoID, for the even-numbered target videos the category label was stored on a map and dumped to a JSON file named ‘category_map.json’. The function ‘target_videos_features’ is defined to store the features for all the target videos these entries are stored in the database as well as a JSON file named ‘total_target_features.json’. Similarly, for non-target videos, function ‘non_target_videos_features’ extracts all the features from the videos and stores them in a JSON named ‘total_non_target_features.json’. The JSON files that are created as a part of this task are available in the database directory.

As a part of feature extraction, the task is to register a hook for layers namely layer3, layer4, and avgpool layers of the r3d_18 pre-trained network. For a given video file that is part of an action, the video frames are gathered for the specific video using OpenCV, and the videos that are being processed are captured to visualize for better human readability. Read the entire videos and append them to a list to get the total number of frames if frames are less than 32 the respective videos are skipped. Once the entire video is processed, we then need to select 32 video frames from the video frames and skip the videos that have frames less than 32. Selecting the first 32 frames causes various issues such as the action not being captured which may result in gathering the video frames that are not required and eventually don’t contain any features. Here, one of the issues that needs to be addressed is the problem with selecting of frames the frames that are selected should possess the action of the video. But to obtain this, we need an approach that selects the frames from the entire video (OpenAI, 2024). To make this happen, a sliding window approach was chosen (You et al., 2019) where the window size was kept to 32. From each of the 32 frames, the frames are gathered. Once the output is gathered from all the slides, the transformation is applied that resizes the frames and stacks them to get the required channels. Since the sliding window approach has many video frames depending on the total frames. From the processed frames max-pooling is applied to gather the maximum values from the frames and then take the mean from it to obtain a 512 size of tensors.

The function 'create_cluster_representatives' is used to create the respective 40 cluster representatives for each value pair of sigma and tau. It selects the top 400 highest-confidence STIP values for each nontarget video and takes a random sample of 10000 for every sigma-tau value and applies KMeans, to obtain the necessary cluster representatives. The function 'get_HoG_HoF_Features' is used to extract the HoG and HoF features from STIPs. Firstly, we select and load the first 400 highest-confidence STIPS by iterating over the predefined values which are namely tau and sigma. These are appended to the corresponding feature list that is HoG and HoF. We use the K-Means clustering technique to assign the STIP to the closest cluster representatives of HoG and HoF. We find the closest distance between the clusters using Euclidean distance. Since HoG and HoF are continuous vectors, Euclidean distance effectively captures the variations in the data also, computationally Euclidean distance is effective. The closest clusterIDs are stored. For the clusterID's, that were computed histograms to see how the features fall under each cluster. Once this process is done for all the tau and sigma values two separate lists were made for HoG and HoF feature descriptors. There are a few STIPS that do not have any value for a particular sigma-tau pair, in this case, zeroes are concatenated as the closest cluster.

### 5.1.2 Task - 0b:

Implemented a main function, that would take video name or videoID, feature space, and a value 'm' that lists and visualizes the top m similar target videos that have even-numbered videoID. If given a videoID, the video name that corresponds to that specific videoID is extracted and shared as a parameter to that respective function. The specific features for a given video are taken from JSON if the feature space is layer_3, layer_4, and avgpool. If the feature space is HoG and HoF the values are extracted from the database. To the extracted features, the target features JSON is loaded and its corresponding videoID is used to check the similarity with the query video that is provided. The results are sorted based on the distance and the top m videos along with their scores are printed. The similarity between the query video and the even-numbered target video is calculated using Euclidean distance. As the distance measure between the closest videos would be near zero, Euclidean given the scenario between actions. Even though the Manhattan Distance was calculated the similar videos listed are not significantly correct when compared with Euclidean distance. As a result, Euclidean was preferred to Manhattan distance.

## 5.2 Task - 1:

As a preprocessing for Task-1, the features of the target videos are loaded along with the category map from the JSON. A function named, 'feature_selection' is defined which makes a matrix of all the features for a class label and takes a mean for the entire features for a specific label. Similarly, this is made for all class labels. The mean features that are calculated for each class label are stored in a JSON. This is made so that the comparison with a given video feature and the query feature can be calculated. Then, for a given video name and the feature space, the feature vector for the query video would be extracted. The feature for the given video is gathered and the distance between each class label feature to the query video feature is calculated using the cosine similarity as a distance measure and the top ‘l’ labels are displayed. Choice of Cosine Similarity is that the same distance calculation was made with the Manhattan distance but the labels are not being predicted properly.

**5.3 Task - 2**:
As part of this task, the users are prompted to select the feature space, value (s) that defines the number of latent semantics, and a dimensionality reduction technique namely, PCA, SVD, LDA, and KMeans. Depending on the dimensionality reduction the following methods are adapted for the computation of ‘s’ latent semantics.

**PCA:**
For the implementation of PCA, firstly, we create a user-defined function whose input parameters are all the video features in the feature space, the number of principal components that one wants to retain, and the feature space. We calculate the covariance matrix from the input data by using the numpy library, which shows features varying together and shows a relation between them in the form of a matrix. We then compute the eigenvectors and the eigenvalues of the covariance matrix that we got above; the eigenvalues here indicate the variance that each principal component captures. Then, these eigenvalues and their corresponding eigenvectors need to be sorted in descending order to focus on the most important components. It selects the top ‘s’ latent eigenvalues and matching eigenvectors to form the left matrix, the principal components, and a core matrix, filled with the chosen eigenvalues on a diagonal of an otherwise empty matrix. Using these eigenvectors, projection of the original data into a lower-dimensional space is performed, resulting in the PCA-reduced data. A dictionary that is JSON-compatible is also made which holds the projected data points as key-value pairs. It prints top eigenvalues for reference at the end and saves the left matrix, core matrix, and JSON data in CSV and JSON, respectively, for further analysis or visualization.

**SVD:**
For SVD, we define a function that takes the same three input parameters as PCA: data, latent_count, and feature space. The SVD process initiates with the calculation of the product of the transpose of the data and the actual data into the matrix '$D^TD$'. To extract the principal components, eigenvalues, and eigenvectors for this matrix get computed. These computed eigenvalues along with their respective eigenvectors will be sorted in descending manner to focus on the most important components. Then, it keeps only the largest eigenvalues, followed by the computation of singular values as the square roots of these eigenvalues. In continuation, this function computes the product of the data and its transpose to make the matrix '$DD^T$' and calculates the eigenvalues and their eigenvectors for this matrix. The eigenvalues and eigenvectors are sorted in the same manner. The eigenvectors of the $DD^T$ matrix make the left matrix, and the eigenvectors of the $D^TD$ matrix form the right matrix. The core matrix is a diagonal matrix containing the top singular values. Then, the data is projected into a reduced-dimensional space by multiplying the data with the selected right singular vectors ‘V’ and core

matrix ‘Σ’. A JSON-compatible dictionary was created to store resulting SVD data points as key-value pairs. We save the Left Singular vectors, core matrix, right singular vectors, and SVD-reduced data into respective CSV and JSON files for further analysis or visualization.

**LDA:**

We create a user-defined function to perform the function of Latent Dirichlet Allocation. It takes three parameters: data - a 2D array where every sublist corresponds to the feature values of a document; k -the number of topics to identify; and the feature space. First, the LDA process converts the numeric data into string format with space-separated strings for each document. Then, it utilizes the function of the natural language toolkit to download the necessary tokenizer resources and tokenizes these strings into lists of individual words. This is followed by the conversion of the documents to a dictionary representation using libraries of the gensim package, whereby each unique word is assigned an integer identifier. These are then used to create a bag-of-words corpus that is fed into the LDA model. The model is created to find the desired number of topics over a preset number of passes - the number here is set at 15 due to the size of the documents and no visible improvement in the latent semantics created. It then pulls the model-created topic distribution for each document, and a helper function converts the output into organized vectors representing the probability of each document being about every topic. These topic vectors are stored in a JSON-compatible dictionary. The code prints out the found topics by showing the most relevant words to the user for better insight and saves both the topics of the model and the document-topic distributions into JSON files. The model and the dictionary are also saved in the Outputs directory to be used for the following Tasks.

**KMeans:**

As part of this task, the users would provide the feature space. Depending upon the feature space, a feature matrix is made for all the even-numbered target videos, and the videoID is appended so that the videoID is mapped to the features. After that, from scikit-learn, KMeans is imported and an instance of the KMeans is made. Before feeding the values to the KMeans instance to fit, the feature matrix was transformed into a numpy array. The main choice of this is it would be a bit easier to feed the features to the KMeans instance rather than as a list. The transformed feature matrix is fit using Kmeans and the cluster centers are gathered. Once the cluster centers are formed, they are transformed into a list and appended into a list. The next task is to compute the distance from each even-numbered target video to all of the cluster centers and find the minimum value among all of them. Once the minimum values are obtained, make a dictionary of videoID-weight pairs. For a given video, the distance from all of the centers to a specific video was computed using Euclidean distance. The choice of Euclidean is that for the given video, if the video is closest to the center or belongs to that specific cluster the distance is close to zero which means that the video represents the cluster center. Because of this reason, I have used Euclidean distance. Once, the videoID to weight pairs are made, these values are dumped into the JSON. Along with this, another JSON named “cluster_centres.json” was made that has the values of all the cluster centers that are of the form lists. As these cluster centers are used as part of the Task-3. After the computation of the clusters, as an output, the clusters are printed on the terminal.

**5.4 Task - 3**:
For Task-3, the user provides a videoID or a video name which would become the query video. Along with that, a parameter 'm' was taken to list the top 'm' similar videos. If the user wanted to find a similarity that belongs to the feature spaces 'layer3', 'layer4', 'avgpool', 'HOG', and 'HOF', the implementation is the same as Task-0b. If the users specify the latent semantics, the methodology will be discussed below. Using the tensors, a function initiates by loading the features from the JSON files from both target and non-target videos. First, the query video is searched in target data, and later in non-target data if not found in the first one. Once the features are found, the appropriate layer is chosen based on the provided layer_number. Then the Euclidean distance is computed between the layer features of the query video with those of all the even-numbered target videos. The result is sorted by distance, providing the 'l' most similar videos along with the distances. The BoF function interacts with the SQL database to get the feature representation of HOG, and HOF features. The function retrieves the user-specified features and gathers all the features for the target videos that have even Video ID. The Euclidean distance is calculated between the query video and those of the target videos. They are sorted according to the similarity score and then displayed. The "kmeans_similarity" function first loads the cluster centers from the JSON file that represents the clustering of video features. It gets the relevant features from the input query video based on the specified layer numbers. Euclidean distance is used to compute the features of the input query video and the cluster center, providing the comparison of similarity with the existing clusters. The user provided the visualization using the OpenCV technique. If the user wants to visualize any more videos, an option to do that is also provided.

For PCA, we use the Left matrix obtained from the last task to reduce the feature space of the given input video and compare it using Euclidean distance with all the videos' reduced feature space. For SVD, we use the Right Matrix to obtain the required dimensionality. For LDA, we use the LDA model trained in Task-2 and the dictionary. We tokenize the input vector and convert it to a bag-of-words (BoW) using the dictionary. We get the topic distribution using the trained model.

For all of the dimensionality reduction techniques, a prompt was made to gather the feature space from the user. For the query video, the respective feature was extracted. As a part of K-means, the cluster centers that are formed as a part of Task-2 are used and read from the JSON. Euclidean distance was used to measure the distance from the cluster centers to the feature space. This would make a list of features that is of length number of cluster centers (that is the latent semantics). Then for all the videos in the target dataset, that respective feature was extracted. For the extracted feature distance was calculated with the cluster centers which makes a list that is of equal length to the query video. By doing so, make the features of the query video and the videos in the target data set of the same length to compute the similarity. Now, the features that are generated for each target video, and the distance was computed with the query video. The video name and the distance are appended to a list. Taking distance as a metric the videos are sorted in ascending order. The main use of Euclidean distance is better results for most of the target videos observed.

### 5.5 Task - 4:

For this specific task, a preprocessing step was made that uses, to store the feature for each label from the target videos. Through this implementation, it was assumed to consider only the target video label. As a preprocessing step, for each of the target video labels, a feature matrix was made which includes all the feature spaces. The method adopted for K-Means is, for an action and a feature space, all the features are gathered. For the gathered features, K-means was employed with cluster centers that were computed as part of Task 2. So, for these features distance was calculated from the centroids to the feature of all the target videos. A json is made, that represents the action label to feature matching that would be in the database directory. For the extracted clusters, the feature distance from the centroids to all the videos was calculated to reduce the dimension. To compute the distance between cluster centroids and the feature spaces, Euclidean distance was used. As the features that are similar to each other Euclidean would give values near to zero. Now, this process was repeated for all of the action labels and the features. All these features are stored in a JSON. Then for each of the actions, the mean of the features was taken which would represent the action feature of dimension 1*s (s-latent semantics). Similarly, this process was used for all the features and all the target videos. Since, the computation needed to be done for all the target videos, the user was prompted for the feature space the similarity should be calculated. The methodology was, for each of the target videos the features were extracted and to the cluster centers the distance was calculated using Euclidean. After this, the distance between the reduced feature and query video was calculated using Euclidean and the top m similar videos were printed. For the calculation of the similar videos and calculation of the reduced feature for a given video, the distance try was made with Euclidean, Manhattan and Cosine Similarity. At the end, Euclidean was able to list top m similar videos in a better way than all other features.

For PCA, we get the average of the features stored in the array using the predefined function inside the numpy library. For LDA, we use the LDA model trained in task 2 and the dictionary. We tokenize the input string, and convert it to a bag-of-words(BoW) using the dictionary. We get the topic distribution using the trained model. According to the requirement of the user, the PCA or LDA function is invoked. The user needs to tell which latent feature space needs to be used, and accordingly, the final result is obtained.

## 6. Interface Specification:

The goal of this project is to use the 6 feature models that are part of Phase-1 and these are revised for Phase-2. The features that are gathered are used to gather the latent features using various approaches. In order to keep the interface simple, a terminal way of approach is chosen where in the terminal the code is executed and the user provides input like a video name. Depending on the task the similar video scores are printed in the terminal.

### 6.1 Obstacle Description:

*Video data complexity*: Compared to image-based feature extraction, video feature extraction is more complex due to the combination of spatial (static object) and temporal (movement) information. A successful solution must take into account both.

*Efficiency and Accuracy*: It can be computationally costly to extract features from a large number of frames. The system must strike a compromise between feature extraction accuracy and computational efficiency.
*Feature Representation*: A thorough representation of the video content may not be possible with basic feature extraction techniques. A combination of HoG, HoF, and deep learning-based features is required to achieve better accuracy.
*Similarity Detection*: Following feature extraction, an effective technique for distinguishing between and determining shared elements across videos is needed. Defining videos in a similar feature space that captures their time-based and spatial features is the difficult part.

## 7. System requirements/installation and execution Instructions:

**7.1 System Requirements:**

1. CPU: 8 Core CPU
2. RAM: 8GB
3. GPU: Not Essentially used.

**7.2 SOL Requirements:**

1. Nodes: 2 (Jupyter Notebook)

**7.3 Programming Language:**

1. Python

**7.4 Modules Required:**

1. OS
2. JSON

**7.5 Libraries:**

1. Pytorch
2. OpenCV
3. Scikit-learn
4. Numpy
5. CSV
6. Matplotlib
7. Av
8. Prettytable
9. Gensim
10. Nltk
11. Sqlite3

## 8. Execution Instructions:

As a prerequisite, the dataset being a large file size, as part of the submission process, the directory structure along with the code would be supplemented. The users are requested to download the dataset from [2] and place it in a directory named ‘hmdb51_org’. Similarly, from [2], download the STIP features for the HMDB51 dataset and place them in ‘hmdb51_stips’. During the execution of the tasks, the population of the database and output directories would happen.

The ‘Code’ directory consists of all the code for this respective phase which are all Python files. To execute them, the command used is ‘python3 <filename>’. The execution description for each of the tasks is described below.

**Task-0a:** The python file ‘Task_0a.py’ contains all the functions that correspond to the Task-0a for this phase. Execute the Python file from the terminal using ‘python3 Task_0a.py’. This task creates json files namely ‘videoID.json’, ‘category_map.json’, ‘total_target_features.json’, and ‘total_non_target_features.json’ in the database directory.
**Task-0b:** The Python file ‘Task_0b.py’ contains the functions that implement Task-0b. Execute it using ‘python3 Task_0b.py’. After that the users are prompted to which way they would like to provide the input video, that is the users need to select 1 for video file name and 2 for videoID. Depending on that info, the users need to enter a video name or videoID. Once this is done, a prompt is made to select the video feature that the user wants to use to find similar videos, depending on the feature space we visualize the ‘m’ similar videos. After that, a prompt was made to select the visualization technique which is using OpenCV, HoG, and HoF.

**Task-1:** Firstly, as a pre-processing step, execute ‘Task_1_preprocess.py’ using the command ‘python3 Task_1_preprocess.py’. This essentially creates a JSON named ‘feature_label_representation.json’ and ‘HoG_HoF_feature_label_representation.json’. After that, execute ‘python3 Task_1.py’, after which the users are prompted to which way they would like to provide the input video, that is the users need to select 1 for the video file name and 2 for videoID. Depending on that info, the users need to enter a video name or videoID. Once this is done, a prompt is made to select the feature space that the user wants to use to find similar labels, depending on the feature space we visualize the ‘l’ similar labels.

**Task-2:** The user needs to execute the python script named ‘Task_2.py’ and execute it using the command ‘python3 Task_2.py’ which prompts the user to select the feature, ‘s’ semantics, a dimensionality reduction technique namely ‘PCA’, ‘SVD’, ‘LDA’, ‘KMeans’. After the computation of ‘s’ semantics, they were printed on the terminal window. And the output for these specific tasks is made in a directory named ‘Output’.

**Task-3:** As a part of this task, the user needs to execute the Python script name ‘Task-3.py’ and execute the command ‘python3 Task_3.py’. Prompts the user to enter a video name or videoID. A prompt would be made to select the user-selected feature space or latent semantics. In addition to this, takes input for value ‘m’ and feature space used as part of Task-2. Depending upon the feature spaces and the latent semantics, the ‘m’ most similar videos are printed. After this, a prompt would be made to visualize the printed videos.

**Task-4:** For Task-4, the user needs to execute the python script using the command ‘python3 Task_4.py’. The users are prompted to enter the video label which is the target video labels and latent semantics. Also, a value for ‘m’ to list ‘m’ similar videos. Also, the users are prompted to enter the feature space which was selected as part of Task-2.

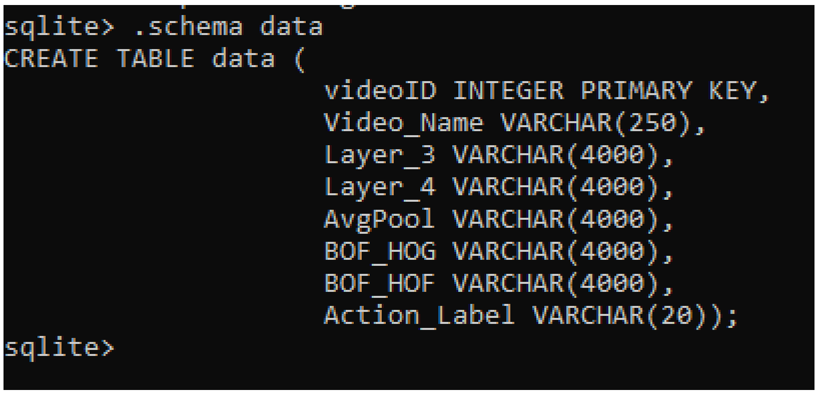
## 8. Related Work:

In (Yoon & Han, 2022), the authors of this paper worked on 'Content Based Video Retrieval', to extract the features models the authors utilized 3D-CNN as their pre-trained neural network, the main choice of this approach as this would benefit the developers when working on limited resources. But even, through this approach when dimensionality reduction happens there would be a feature loss. So, the authors used network representation (Snell, Swersky, & Zemel, 2017) to eliminate feature loss.

## 9. Results:

**Task-0:**

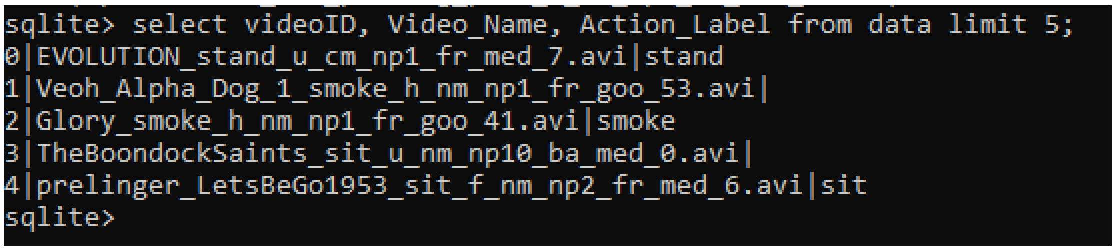
**Task-0a:**



```
sqlite> .schema data
CREATE TABLE data (
                        videoID INTEGER PRIMARY KEY,
                        Video_Name VARCHAR(250),
                        Layer_3 VARCHAR(4000),
                        Layer_4 VARCHAR(4000),
                        AvgPool VARCHAR(4000),
                        BOF_HOG VARCHAR(4000),
                        BOF_HOF VARCHAR(4000),
                        Action_Label VARCHAR(20));
sqlite>
```

Fig 1: Picture of the database schema

Fig-1 shows us the schema of the SQL table created to store the features using the video ID as the primary key. The features from Phase 1 Task 1: Layer_3, Layer_4, AvgPool, BOF_HOG, BOF_HOF, and their corresponding action labels are stored. They are stored according to the video names in the table.



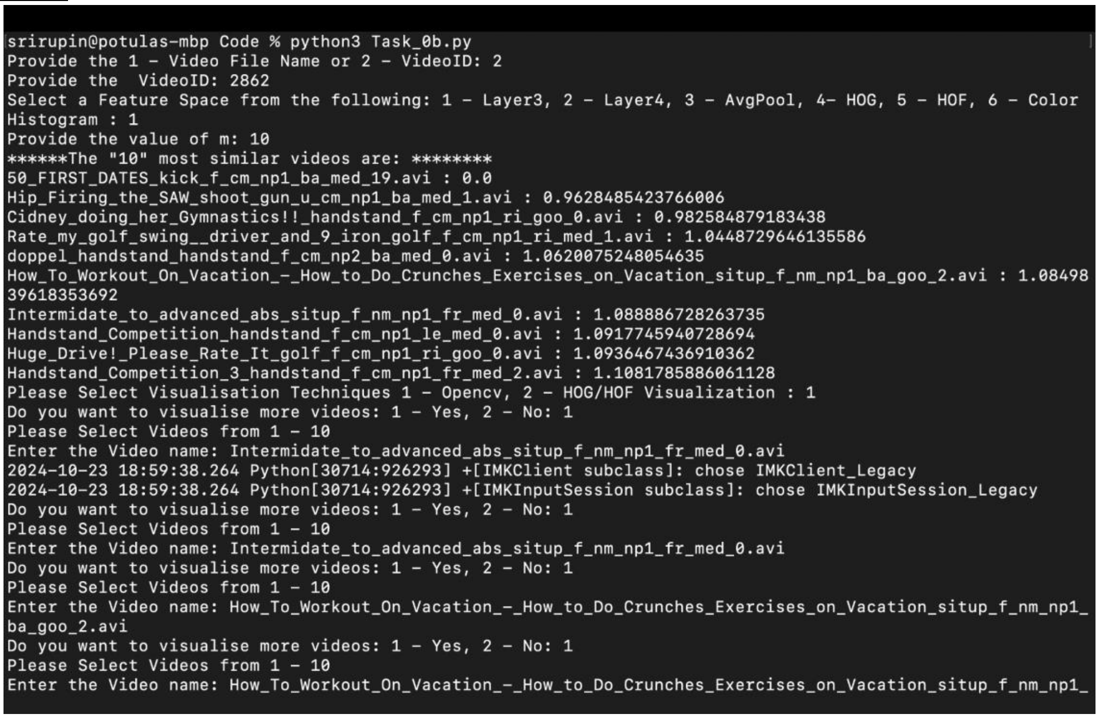
```
sqlite> select videoID, Video_Name, Action_Label from data limit 5;
0|EVOLUTION_stand_u_cm_np1_fr_med_7.avi|stand
1|Veoh_Alpha_Dog_1_smoke_h_nm_np1_fr_goo_53.avi|
2|Glory_smoke_h_nm_np1_fr_goo_41.avi|smoke
3|TheBoondockSaints_sit_u_nm_np10_ba_med_0.avi|
4|prelinger_LetsBeGo1953_sit_f_nm_np2_fr_med_6.avi|sit
sqlite>
```

*Fig 2: The above picture describes the videoID, Video Name, and Action Label.*

Figure-2 shows the top 5 rows of the SQL table, which are arranged in ascending order by their videoID. The odd target videos would not have the action label along with them as per the instructions provided. This provides a snippet of the entire database.

**Task-0b:**

**Input -1:**



```
srirupin@potulas-mbp Code % python3 Task_0b.py
Provide the 1 - Video File Name or 2 - VideoID: 2
Provide the  VideoID: 2862
Select a Feature Space from the following: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color
Histogram : 1
Provide the value of m: 10
******The "10" most similar videos are: ********
50_FIRST_DATES_kick_f_cm_np1_ba_med_19.avi : 0.0
Hip_Firing_the_SAW_shoot_gun_u_cm_np1_ba_med_1.avi : 0.9628485423766006
Cidney_doing_her_Gymnastics!!_handstand_f_cm_np1_ri_goo_0.avi : 0.982584879183438
Rate_my_golf_swing__driver_and_9_iron_golf_f_cm_np1_ri_med_1.avi : 1.0448729646135586
doppel_handstand_handstand_f_cm_np2_ba_med_0.avi : 1.0620075248054635
How_To_Workout_On_Vacation_-_How_to_Do_Crunches_Exercises_on_Vacation_situp_f_nm_np1_ba_goo_2.avi : 1.08498
39618353692
Intermidate_to_advanced_abs_situp_f_nm_np1_fr_med_0.avi : 1.088886728263735
Handstand_Competition_handstand_f_cm_np1_le_med_0.avi : 1.0917745940728694
Huge_Drive!_Please_Rate_It_golf_f_cm_np1_ri_goo_0.avi : 1.0936467436910362
Handstand_Competition_3_handstand_f_cm_np1_fr_med_2.avi : 1.1081785886061128
Please Select Visualisation Techniques 1 - Opencv, 2 - HOG/HOF Visualization : 1
Do you want to visualise more videos: 1 - Yes, 2 - No: 1
Please Select Videos from 1 - 10
Enter the Video name: Intermidate_to_advanced_abs_situp_f_nm_np1_fr_med_0.avi
2024-10-23 18:59:38.264 Python[30714:926293] +[IMKClient subclass]: chose IMKClient_Legacy
2024-10-23 18:59:38.264 Python[30714:926293] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
Do you want to visualise more videos: 1 - Yes, 2 - No: 1
Please Select Videos from 1 - 10
Enter the Video name: Intermidate_to_advanced_abs_situp_f_nm_np1_fr_med_0.avi
Do you want to visualise more videos: 1 - Yes, 2 - No: 1
Please Select Videos from 1 - 10
Enter the Video name: How_To_Workout_On_Vacation_-_How_to_Do_Crunches_Exercises_on_Vacation_situp_f_nm_np1_
ba_goo_2.avi
Do you want to visualise more videos: 1 - Yes, 2 - No: 1
Please Select Videos from 1 - 10
Enter the Video name: How_To_Workout_On_Vacation_-_How_to_Do_Crunches_Exercises_on_Vacation_situp_f_nm_np1_
```

*Fig 3: 'm' most similar videos found*

The above figure(Fig-3) shows the demonstration of the program where layer 3 is used to find the similar videos. We test it out for m=10 values. After getting the 10 most similar videos, an option is provided to know in what way the visualization needs to be done. Option of both OpenCV and HOG/HOF are provided to the user. Furthermore, an option is provided to show the video among the most similar videos which are found before. 'Fig-4' shows the visualization of the video "Intermediate_to_advanced_abs_situp_f_nm_np1_fr_med_0.avi". This is how task 0b can be performed using the layers as the feature chosen to find the most similar videos. The given video is of action 'kick'. However, in the top m similar videos that weren't observed. But the feature distance from the same image is zero as observed.

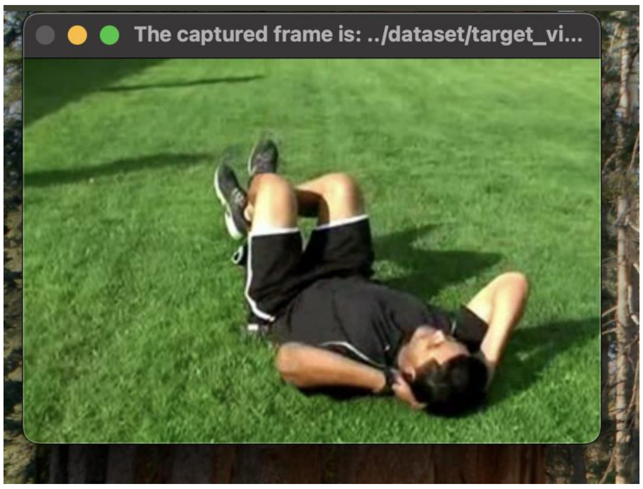
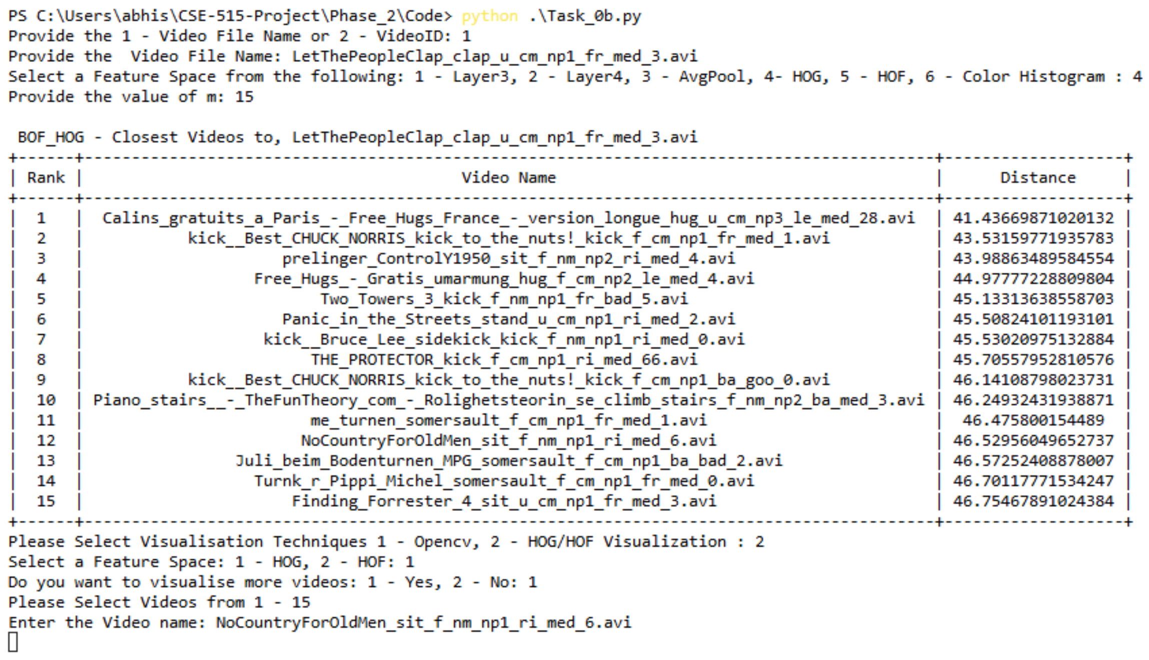


*Fig 4: Visualization of video*

## Input-2:



```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_0b.py
Provide the 1 - Video File Name or 2 - VideoID: 1
Provide the  Video File Name: LetThePeopleClap_clap_u_cm_np1_fr_med_3.avi
Select a Feature Space from the following: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram : 4
Provide the value of m: 15

 BOF_HOG - Closest Videos to, LetThePeopleClap_clap_u_cm_np1_fr_med_3.avi
+------+------------------------------------------------------------------------------------------+--------------------+
| Rank |                                        Video Name                                        |      Distance      |
+------+------------------------------------------------------------------------------------------+--------------------+
|  1   |   Calins_gratuits_a_Paris_-_Free_Hugs_France_-_version_longue_hug_u_cm_np3_le_med_28.avi  | 41.43669871020132 |
|  2   |            kick__Best_CHUCK_NORRIS_kick_to_the_nuts!_kick_f_cm_np1_fr_med_1.avi           | 43.53159771935783 |
|  3   |                     prelinger_ControlY1950_sit_f_nm_np2_ri_med_4.avi                      | 43.98863489584554 |
|  4   |                  Free_Hugs_-_Gratis_umarmung_hug_f_cm_np2_le_med_4.avi                    | 44.97777228809804 |
|  5   |                         Two_Towers_3_kick_f_nm_np1_fr_bad_5.avi                           | 45.13313638558703 |
|  6   |                     Panic_in_the_Streets_stand_u_cm_np1_ri_med_2.avi                      | 45.50824101193101 |
|  7   |                   kick__Bruce_Lee_sidekick_kick_f_nm_np1_ri_med_0.avi                     | 45.53020975132884 |
|  8   |                        THE_PROTECTOR_kick_f_cm_np1_ri_med_66.avi                          | 45.70557952810576 |
|  9   |            kick__Best_CHUCK_NORRIS_kick_to_the_nuts!_kick_f_cm_np1_ba_goo_0.avi           | 46.14108798023731 |
|  10  | Piano_stairs__-_TheFunTheory_com_-_Rolighetsteorin_se_climb_stairs_f_nm_np2_ba_med_3.avi | 46.24932431938871 |
|  11  |                        me_turnen_somersault_f_cm_np1_fr_med_1.avi                         |  46.475800154489  |
|  12  |                       NoCountryForOldMen_sit_f_nm_np1_ri_med_6.avi                        | 46.52956049652737 |
|  13  |                 Juli_beim_Bodenturnen_MPG_somersault_f_cm_np1_ba_bad_2.avi                | 46.57252408878007 |
|  14  |                  Turnk_r_Pippi_Michel_somersault_f_cm_np1_fr_med_0.avi                    | 46.70117771534247 |
|  15  |                        Finding_Forrester_4_sit_u_cm_np1_fr_med_3.avi                      | 46.75467891024384 |
+------+------------------------------------------------------------------------------------------+--------------------+
Please Select Visualisation Techniques 1 - Opencv, 2 - HOG/HOF Visualization : 2
Select a Feature Space: 1 - HOG, 2 - HOF: 1
Do you want to visualise more videos: 1 - Yes, 2 - No: 1
Please Select Videos from 1 - 15
Enter the Video name: NoCountryForOldMen_sit_f_nm_np1_ri_med_6.avi
```

*Fig 5:HoG Distance for a non-target video*

The above figure(Fig-5) showcases the usage of HoG features to find the 'm' most similar videos. 'm' is specified as 15 here. The distance measured is Euclidean distance for measuring the similarity of the videos. 'Fig-6' shows the histogram plots of the HoG features which are for the above. Since, the video file name that was provided being from non target videos and the search is on target videos, the first video would not be at a distance of 0. The given video input video is of type clap but as part of 'm' similar videos we could not observe the videos of action type clap.

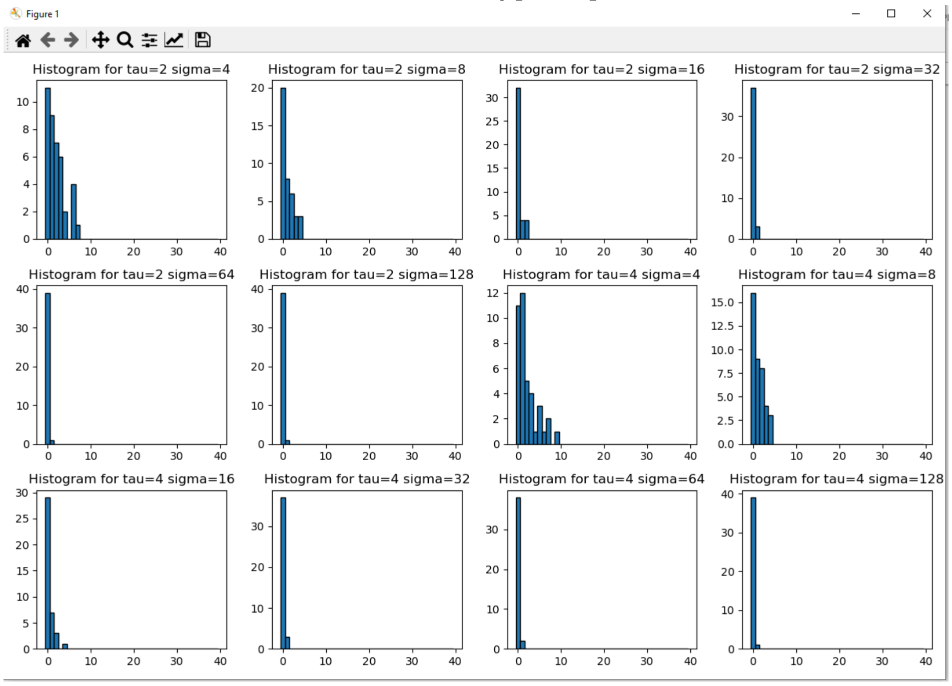
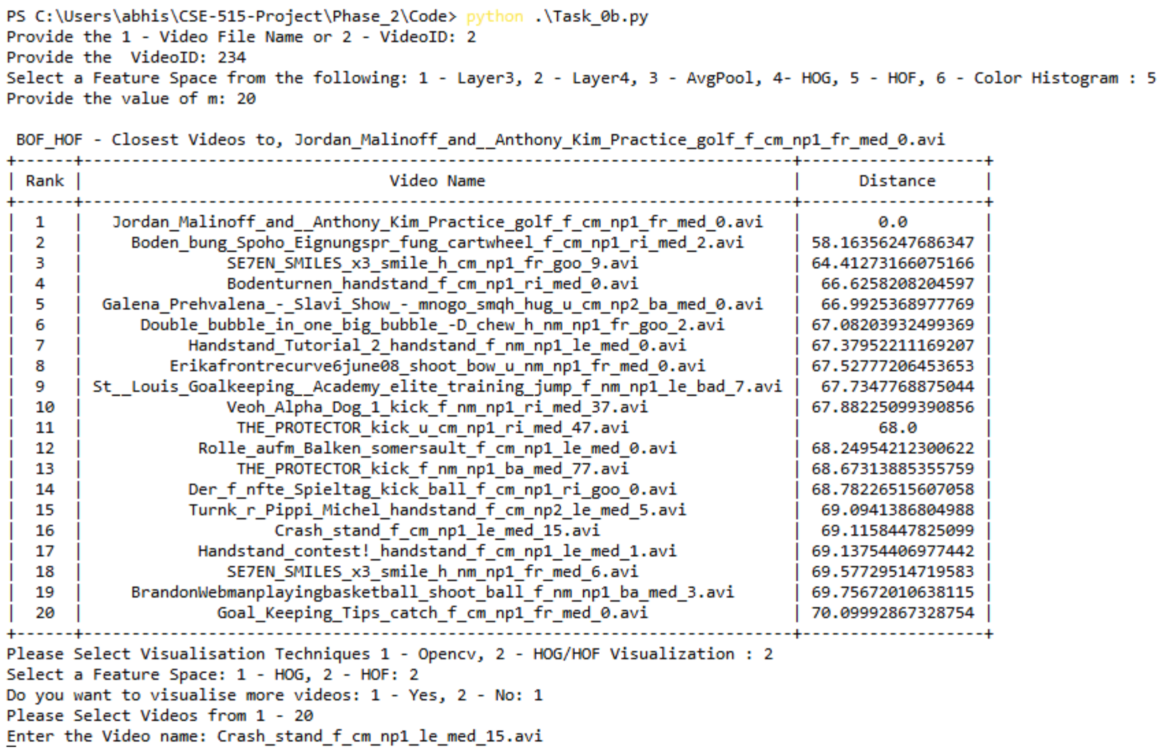


*Fig 6: Histograms for the various values of Tau and Sigma*

## Input-3: Target Video HoF



```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_0b.py
Provide the 1 - Video File Name or 2 - VideoID: 2
Provide the  VideoID: 234
Select a Feature Space from the following: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram : 5
Provide the value of m: 20

 BOF_HOF - Closest Videos to, Jordan_Malinoff_and__Anthony_Kim_Practice_golf_f_cm_np1_fr_med_0.avi
+------+-------------------------------------------------------------------------+-------------------+
| Rank |                               Video Name                                |      Distance     |
+------+-------------------------------------------------------------------------+-------------------+
|  1   |    Jordan_Malinoff_and__Anthony_Kim_Practice_golf_f_cm_np1_fr_med_0.avi    |        0.0        |
|  2   |      Boden_bung_Spoho_Eignungspr_fung_cartwheel_f_cm_np1_ri_med_2.avi      | 58.16356247686347 |
|  3   |               SE7EN_SMILES_x3_smile_h_cm_np1_fr_goo_9.avi                 | 64.41273166075166 |
|  4   |               Bodenturnen_handstand_f_cm_np1_ri_med_0.avi                 |  66.6258208204597 |
|  5   |   Galena_Prehvalena_-_Slavi_Show_-_mnogo_smqh_hug_u_cm_np2_ba_med_0.avi    |  66.9925368977769 |
|  6   |        Double_bubble_in_one_big_bubble_-D_chew_h_nm_np1_fr_goo_2.avi        | 67.08203932499369 |
|  7   |           Handstand_Tutorial_2_handstand_f_nm_np1_le_med_0.avi            | 67.37952211169207 |
|  8   |         Erikafrontrecurve6june08_shoot_bow_u_nm_np1_fr_med_0.avi          | 67.52777206453653 |
|  9   |  St__Louis_Goalkeeping__Academy_elite_training_jump_f_nm_np1_le_bad_7.avi  |  67.7347768875044 |
|  10  |               Veoh_Alpha_Dog_1_kick_f_nm_np1_ri_med_37.avi                | 67.88225099390856 |
|  11  |                THE_PROTECTOR_kick_u_cm_np1_ri_med_47.avi                  |        68.0       |
|  12  |            Rolle_aufm_Balken_somersault_f_cm_np1_le_med_0.avi             | 68.24954212300622 |
|  13  |                THE_PROTECTOR_kick_f_nm_np1_ba_med_77.avi                  | 68.67313885355759 |
|  14  |           Der_f_nfte_Spieltag_kick_ball_f_cm_np1_ri_goo_0.avi             | 68.78226515607058 |
|  15  |           Turnk_r_Pippi_Michel_handstand_f_cm_np2_le_med_5.avi            |  69.0941386804988 |
|  16  |                    Crash_stand_f_cm_np1_le_med_15.avi                     |  69.1158447825099 |
|  17  |            Handstand_contest!_handstand_f_cm_np1_le_med_1.avi             | 69.13754406977442 |
|  18  |               SE7EN_SMILES_x3_smile_h_nm_np1_fr_med_6.avi                 | 69.57729514719583 |
|  19  |      BrandonWebmanplayingbasketball_shoot_ball_f_nm_np1_ba_med_3.avi       | 69.75672010638115 |
|  20  |              Goal_Keeping_Tips_catch_f_cm_np1_fr_med_0.avi                | 70.09992867328754 |
+------+-------------------------------------------------------------------------+-------------------+
Please Select Visualisation Techniques 1 - Opencv, 2 - HOG/HOF Visualization : 2
Select a Feature Space: 1 - HOG, 2 - HOF: 2
Do you want to visualise more videos: 1 - Yes, 2 - No: 1
Please Select Videos from 1 - 20
Enter the Video name: Crash_stand_f_cm_np1_le_med_15.avi
```

*Fig 7:HoF Distance and Visualization*

'Fig-7' showcases the usage of HoG features to find the 'm' most similar videos. 'm' is specified as 20 here. The distance measured is Euclidean distance for measuring the similarity of the videos. 'Fig-8' shows the histogram plots of the HoG features which are for the above. This shows us the closest 20 videos for the given video ID(videoID:234). The given video is of action-type golf and the first similar video represents the input video.

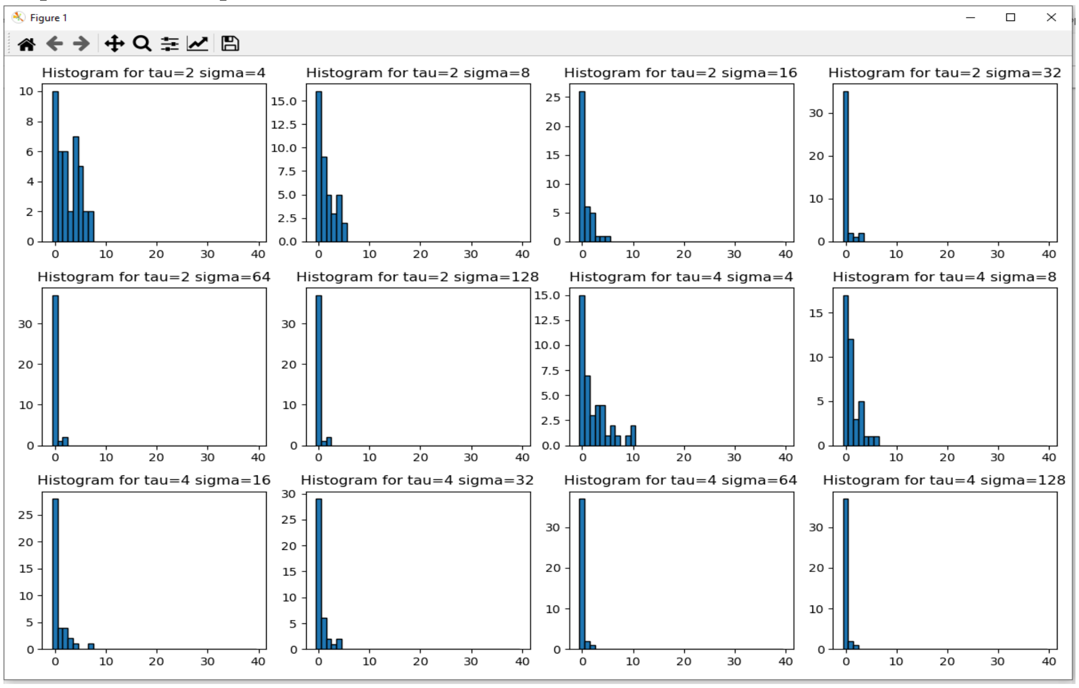
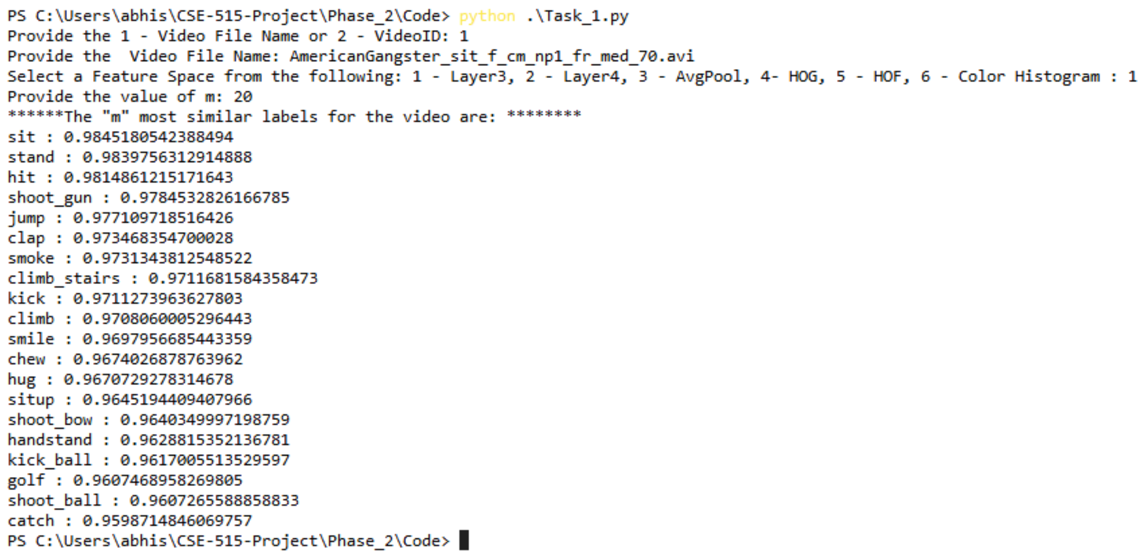


*Fig 8: Histogram outputs for various values of Tau and Sigma*

**Task-1:**

**<u>Input - 1:</u>**

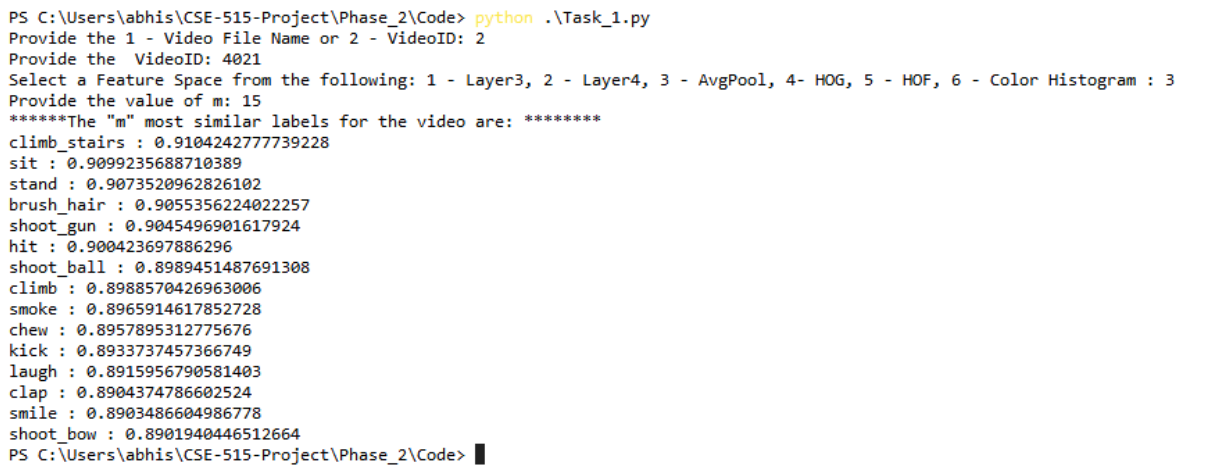


```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_1.py
Provide the 1 - Video File Name or 2 - VideoID: 1
Provide the  Video File Name: AmericanGangster_sit_f_cm_np1_fr_med_70.avi
Select a Feature Space from the following: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram : 1
Provide the value of m: 20
******The "m" most similar labels for the video are: ********
sit : 0.9845180542388494
stand : 0.9839756312914888
hit : 0.9814861215171643
shoot_gun : 0.9784532826166785
jump : 0.977109718516426
clap : 0.973468354700028
smoke : 0.9731343812548522
climb_stairs : 0.9711681584358473
kick : 0.9711273963627803
climb : 0.9708060005296443
smile : 0.9697956685443359
chew : 0.9674026878763962
hug : 0.9670729278314678
situp : 0.9645194409407966
shoot_bow : 0.9640349997198759
handstand : 0.9628815352136781
kick_ball : 0.9617005513529597
golf : 0.9607468958269805
shoot_ball : 0.9607265588858833
catch : 0.9598714846069757
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> █
```

*Fig 9: Most similar video labels using Layer 3*

'Fig-9' shows the 'm' most similar labels for the given input. We have given the Video name as the input and performed it for m=20 as the most similar labels. The similarity score is found on the basis of Euclidean distance. The given video file name is of type sit and in the similar labels we could see that the action type 'sit' was produced.

**<u>Input - 2:</u>**

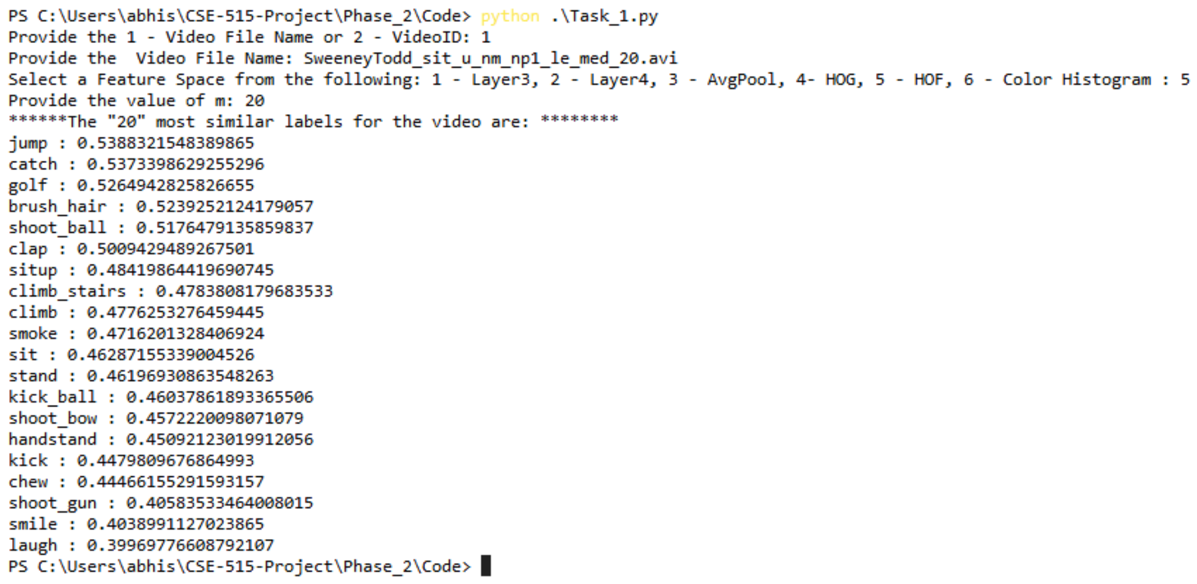


```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_1.py
Provide the 1 - Video File Name or 2 - VideoID: 2
Provide the  VideoID: 4021
Select a Feature Space from the following: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram : 3
Provide the value of m: 15
******The "m" most similar labels for the video are: ********
climb_stairs : 0.9104242777739228
sit : 0.9099235688710389
stand : 0.9073520962826102
brush_hair : 0.9055356224022257
shoot_gun : 0.9045496901617924
hit : 0.900423697886296
shoot_ball : 0.8989451487691308
climb : 0.8988570426963006
smoke : 0.8965914617852728
chew : 0.8957895312775676
kick : 0.8933737457366749
laugh : 0.8915956790581403
clap : 0.8904374786602524
smile : 0.8903486604986778
shoot_bow : 0.8901940446512664
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> █
```

*Fig 10: Most similar video labels using Averagepool*

'Fig-10' shows the 'm' most similar labels for the given input. We have given the Video ID as the input and performed it for m=15 as the most similar labels. The similarity score is found on the basis of Euclidean distance. The given input video name is from nontarget video, since the labels are not available it was not accurately predicted.

## Input - 3:



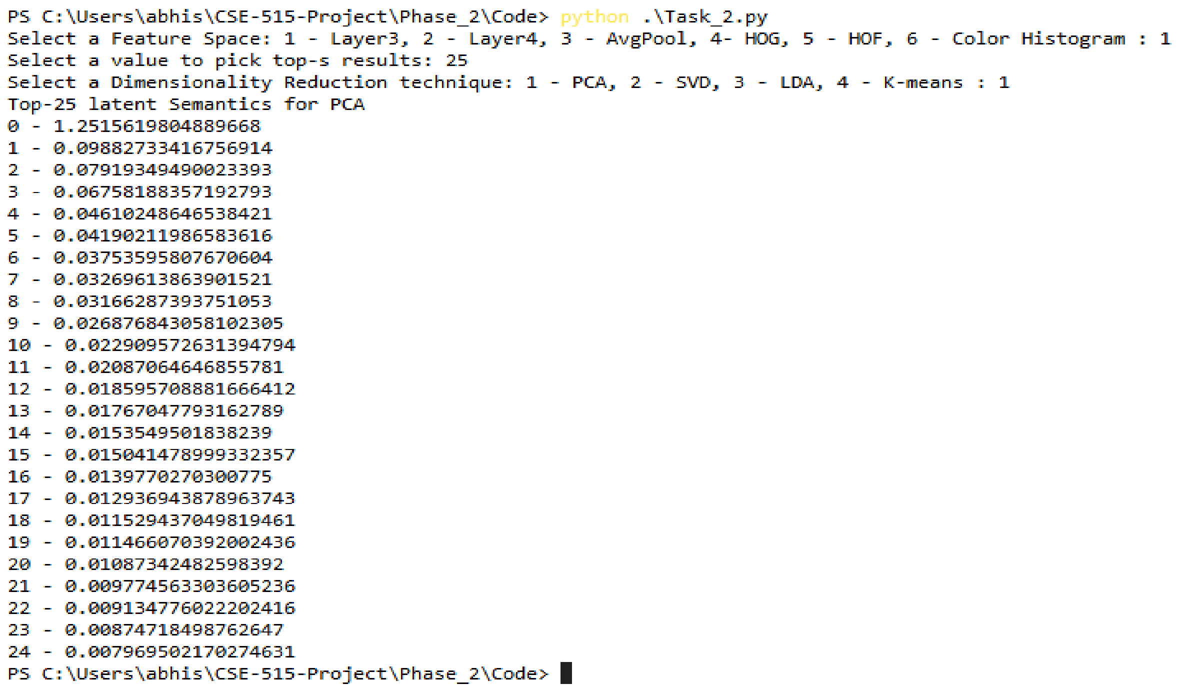
```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_1.py
Provide the 1 - Video File Name or 2 - VideoID: 1
Provide the  Video File Name: SweeneyTodd_sit_u_nm_np1_le_med_20.avi
Select a Feature Space from the following: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram : 5
Provide the value of m: 20
******The "20" most similar labels for the video are: ********
jump : 0.5388321548389865
catch : 0.5373398629255296
golf : 0.5264942825826655
brush_hair : 0.5239252124179057
shoot_ball : 0.5176479135859837
clap : 0.5009429489267501
situp : 0.48419864419690745
climb_stairs : 0.4783808179683533
climb : 0.4776253276459445
smoke : 0.4716201328406924
sit : 0.46287155339004526
stand : 0.46196930863548263
kick_ball : 0.46037861893365506
shoot_bow : 0.4572220098071079
handstand : 0.45092123019912056
kick : 0.4479809676864993
chew : 0.44466155291593157
shoot_gun : 0.40583533464008015
smile : 0.4038991127023865
laugh : 0.39969776608792107
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code>
```

*Fig 11: 'm' most similar labels based on HOF*

'fig-11' shows the most similar labels based on HoF features of the file whose name is given as the input. The value of 'm' taken for this situation is 20. The figure showcases the most similar labels based on the HoF features of each label with respect to the ones the input file is having. For the given video, it was observed that both sit and situp are listed in the similar label predictions. As they both perform similar actions.

## Task-2:

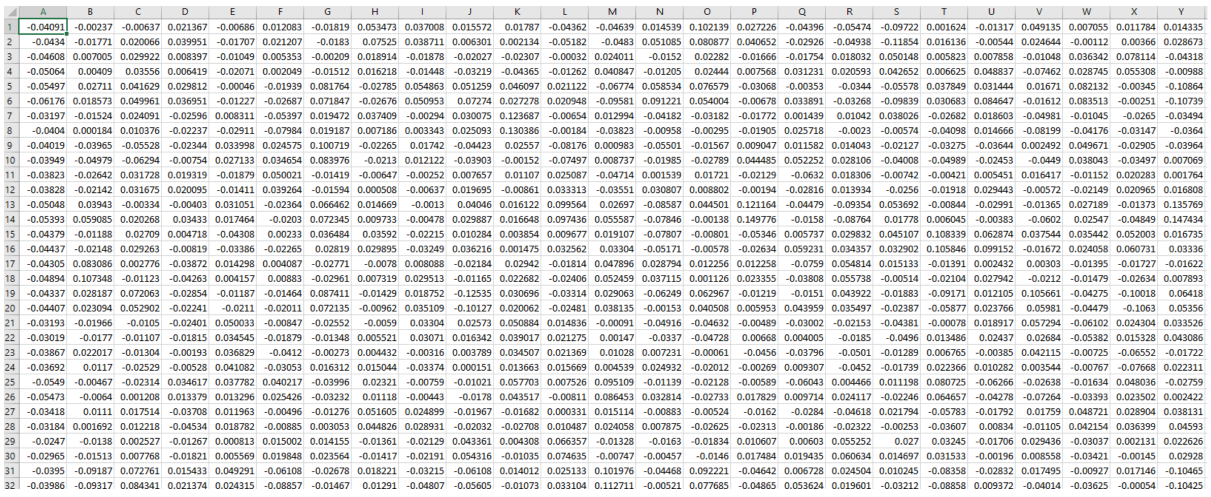
## Input - 1:



```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_2.py
Select a Feature Space: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram : 1
Select a value to pick top-s results: 25
Select a Dimensionality Reduction technique: 1 - PCA, 2 - SVD, 3 - LDA, 4 - K-means : 1
Top-25 latent Semantics for PCA
0 - 1.2515619804889668
1 - 0.09882733416756914
2 - 0.07919349490023393
3 - 0.06758188357192793
4 - 0.04610248646538421
5 - 0.04190211986583616
6 - 0.03753595807670604
7 - 0.03269613863901521
8 - 0.03166287393751053
9 - 0.026876843058102305
10 - 0.022909572631394794
11 - 0.02087064646855781
12 - 0.018595708881666412
13 - 0.01767047793162789
14 - 0.015354950183823 9
15 - 0.015041478999332357
16 - 0.0139770270300775
17 - 0.012936943878963743
18 - 0.011529437049819461
19 - 0.011466070392002436
20 - 0.01087342482598392
21 - 0.009774563303605236
22 - 0.009134776022202416
23 - 0.00874718498762647
24 - 0.007969502170274631
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code>
```

*Fig 12: Latent Semantics using PCA*

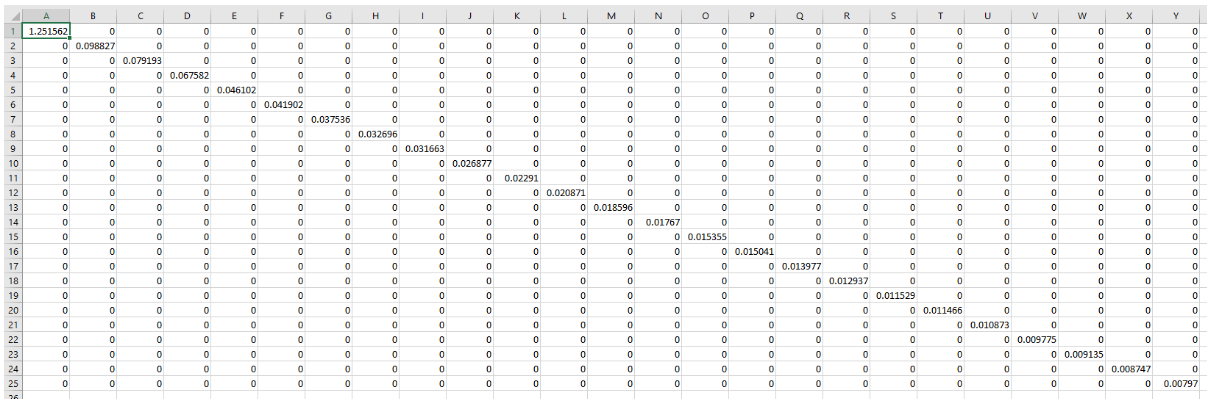
'Fig-12' describes the 's' PCA components of the Layer 3 feature space. It demonstrates how the existing features are converted into 's' latent features. Here, 's' is taken as 25.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -0.04091 | -0.00237 | -0.00637 | 0.021367 | -0.00686 | 0.012083 | -0.01819 | 0.053473 | 0.037008 | 0.015572 | 0.01787 | -0.04362 | -0.04639 | 0.014539 | 0.102139 | 0.027226 | -0.04396 | -0.05474 | -0.09722 | 0.001624 | -0.01317 | 0.049135 | 0.007055 | 0.011784 | 0.014335 |
| 2 | -0.0434 | -0.01771 | 0.020066 | 0.039951 | -0.01707 | 0.021207 | -0.0183 | 0.07525 | 0.038711 | 0.006301 | 0.002134 | -0.05182 | -0.0483 | 0.051085 | 0.080877 | 0.040652 | -0.02926 | -0.04938 | -0.11854 | 0.016136 | -0.00544 | 0.024644 | -0.00112 | 0.00366 | 0.028673 |
| 3 | -0.04608 | 0.007005 | 0.029922 | 0.008397 | -0.01049 | 0.005353 | -0.00209 | 0.018914 | -0.01878 | -0.02027 | -0.02307 | -0.00032 | 0.024011 | -0.0152 | 0.02282 | -0.01666 | -0.01754 | 0.018032 | 0.050148 | 0.005823 | 0.007858 | -0.01048 | 0.036342 | 0.078114 | -0.04318 |
| 4 | -0.05064 | 0.00409 | 0.03556 | 0.006419 | -0.02071 | 0.002049 | -0.01512 | 0.016218 | -0.01448 | -0.03219 | -0.04365 | -0.01262 | 0.040847 | -0.01205 | 0.02444 | 0.007568 | 0.031231 | 0.020593 | 0.042652 | 0.006625 | 0.048837 | -0.07462 | 0.028745 | 0.055308 | -0.00988 |
| 5 | -0.05497 | 0.02711 | 0.041629 | 0.029812 | -0.00046 | -0.01939 | 0.081764 | -0.02785 | 0.054863 | 0.051259 | 0.046097 | 0.021122 | -0.06774 | 0.058534 | 0.076579 | -0.03068 | -0.00353 | -0.0344 | -0.05578 | 0.037849 | 0.031444 | 0.01671 | 0.082132 | -0.00345 | -0.10864 |
| 6 | -0.06176 | 0.018573 | 0.049961 | 0.036951 | -0.01227 | -0.02687 | 0.071847 | -0.02676 | 0.050953 | 0.07274 | 0.027278 | 0.020948 | -0.09581 | 0.091221 | 0.054004 | -0.00678 | 0.033891 | -0.03268 | -0.09839 | 0.030683 | 0.084647 | -0.01612 | 0.083513 | -0.00251 | -0.10739 |
| 7 | -0.03197 | -0.01524 | 0.024091 | -0.02596 | 0.008311 | -0.05397 | 0.019472 | 0.037409 | -0.00294 | 0.030075 | 0.123687 | -0.00654 | 0.012994 | -0.04182 | -0.03182 | -0.01772 | 0.001439 | 0.01042 | 0.038026 | -0.02682 | 0.018603 | -0.04981 | -0.01045 | -0.0265 | -0.03494 |
| 8 | -0.0404 | 0.000184 | 0.010376 | -0.02237 | -0.02911 | -0.07984 | 0.019187 | 0.007186 | 0.003343 | 0.025093 | 0.130386 | -0.00184 | -0.03823 | -0.00958 | -0.00295 | -0.01905 | 0.025718 | -0.0023 | -0.00574 | -0.04098 | 0.014666 | -0.08199 | -0.04176 | -0.03147 | -0.0364 |
| 9 | -0.04019 | -0.03965 | -0.05528 | -0.02344 | 0.033998 | 0.024575 | 0.100719 | -0.02265 | 0.01742 | -0.04423 | 0.02557 | -0.08176 | 0.000983 | -0.05501 | -0.01567 | 0.009047 | 0.011582 | 0.014043 | -0.02127 | -0.03275 | -0.03644 | 0.002492 | 0.049671 | -0.02905 | -0.03964 |
| 10 | -0.03949 | -0.04979 | -0.06294 | -0.00754 | 0.027133 | 0.034654 | 0.083976 | -0.0213 | 0.012122 | -0.03903 | -0.00152 | -0.07497 | 0.008737 | -0.01985 | -0.02789 | 0.044485 | 0.052252 | 0.028106 | -0.04008 | -0.04989 | -0.02453 | -0.0449 | 0.038043 | -0.03497 | 0.007069 |
| 11 | -0.03823 | -0.02642 | 0.031728 | 0.019319 | -0.01879 | 0.050021 | -0.01419 | -0.00647 | -0.00252 | 0.007657 | 0.01107 | 0.025087 | -0.04714 | 0.001539 | 0.01721 | -0.02129 | -0.0632 | 0.018306 | -0.00742 | -0.00421 | 0.005451 | 0.016417 | -0.01152 | 0.020283 | 0.001764 |
| 12 | -0.03828 | -0.02142 | 0.031675 | 0.020095 | -0.01411 | 0.039264 | -0.01594 | 0.000508 | -0.00637 | 0.019695 | -0.00861 | 0.033313 | -0.03551 | 0.030807 | 0.008802 | -0.00194 | -0.02816 | 0.013934 | -0.0256 | -0.01918 | 0.029443 | -0.00572 | -0.02149 | 0.020965 | 0.016808 |
| 13 | -0.05048 | 0.03943 | -0.00334 | -0.00403 | 0.031051 | -0.02364 | 0.066462 | 0.014669 | -0.0013 | 0.04046 | 0.016122 | 0.099564 | 0.02697 | -0.08587 | 0.044501 | 0.121164 | -0.04479 | -0.09354 | 0.053692 | -0.00844 | -0.02991 | -0.01365 | 0.027189 | -0.01373 | 0.135769 |
| 14 | -0.05393 | 0.059085 | 0.020268 | 0.03433 | 0.017464 | -0.0203 | 0.072345 | 0.009733 | -0.00478 | 0.029887 | 0.016648 | 0.097436 | 0.055587 | -0.07846 | -0.00138 | 0.149776 | -0.0158 | -0.08764 | 0.01778 | 0.006045 | -0.00383 | -0.0602 | 0.02547 | -0.04849 | 0.147434 |
| 15 | -0.04379 | -0.01188 | 0.02709 | 0.004718 | -0.04308 | 0.00233 | 0.036484 | 0.03592 | -0.02215 | 0.010284 | 0.003854 | 0.009677 | 0.019107 | -0.07807 | -0.00801 | -0.05346 | 0.005737 | 0.029832 | 0.045107 | 0.108339 | 0.062874 | 0.037544 | 0.035442 | 0.052003 | 0.016735 |
| 16 | -0.04437 | -0.02148 | 0.029263 | -0.00819 | -0.03386 | -0.02265 | 0.02819 | 0.029895 | -0.03249 | 0.036216 | 0.001475 | 0.032562 | 0.03304 | -0.05171 | -0.00578 | -0.02634 | 0.059231 | 0.034357 | 0.032902 | 0.105846 | 0.099152 | -0.01672 | 0.024058 | 0.060731 | 0.03336 |
| 17 | -0.04305 | 0.083086 | 0.002776 | -0.03872 | 0.014298 | 0.004087 | -0.02771 | -0.0078 | 0.008088 | -0.02184 | 0.02942 | -0.01814 | 0.047896 | 0.028794 | 0.012256 | 0.012258 | -0.0759 | 0.054814 | 0.015133 | -0.01391 | 0.002432 | 0.00303 | -0.01395 | -0.01727 | -0.01622 |
| 18 | -0.04894 | 0.107348 | -0.01123 | -0.04263 | 0.004157 | 0.00883 | -0.02961 | 0.007319 | 0.029513 | -0.01165 | 0.022682 | -0.02406 | 0.052459 | 0.037115 | 0.001126 | 0.023355 | -0.03808 | 0.055738 | -0.00514 | -0.02104 | 0.027942 | -0.0212 | -0.01479 | -0.02634 | 0.007893 |
| 19 | -0.04337 | 0.028187 | 0.072063 | -0.02854 | -0.01187 | -0.01464 | 0.087411 | -0.01429 | 0.018752 | -0.12535 | 0.030696 | -0.03314 | 0.029063 | -0.06249 | 0.062967 | -0.01219 | -0.0151 | 0.043922 | -0.01883 | -0.09171 | 0.012105 | 0.105661 | -0.04275 | -0.10018 | 0.06418 |
| 20 | -0.04407 | 0.023094 | 0.052902 | -0.02241 | -0.0211 | -0.02011 | 0.072135 | -0.00962 | 0.035109 | -0.10127 | 0.020062 | -0.02481 | 0.038135 | -0.00153 | 0.040508 | 0.005953 | 0.043959 | 0.035497 | -0.02387 | -0.05877 | 0.023766 | 0.05981 | -0.04479 | -0.1063 | 0.05356 |
| 21 | -0.03193 | -0.01966 | -0.0105 | -0.02401 | 0.050033 | -0.00847 | -0.02552 | -0.0059 | 0.03304 | 0.02573 | 0.050884 | 0.014836 | -0.00091 | -0.04916 | -0.04632 | -0.00489 | -0.03002 | -0.02153 | -0.04381 | -0.00078 | 0.018917 | 0.057294 | -0.06102 | 0.024304 | 0.033526 |
| 22 | -0.03019 | -0.0177 | -0.01107 | -0.01815 | 0.034545 | -0.01879 | -0.01348 | 0.005521 | 0.03071 | 0.016342 | 0.039017 | 0.021275 | 0.00147 | -0.0337 | -0.04728 | 0.00668 | 0.004005 | -0.0185 | -0.0496 | 0.013486 | 0.02437 | 0.02684 | -0.05382 | 0.015328 | 0.043086 |
| 23 | -0.03867 | 0.022017 | -0.01304 | -0.00193 | 0.036829 | -0.0412 | -0.00273 | 0.004432 | -0.00316 | 0.003789 | 0.034507 | 0.021369 | 0.01028 | 0.007231 | -0.00061 | -0.0456 | -0.03796 | -0.0501 | -0.01289 | 0.006765 | -0.00385 | 0.042115 | -0.00725 | -0.06552 | -0.01722 |
| 24 | -0.03692 | 0.0117 | -0.02529 | -0.00528 | 0.041082 | -0.03053 | 0.016312 | 0.015044 | -0.03374 | 0.000151 | 0.013663 | 0.015669 | 0.004539 | 0.024932 | -0.02012 | -0.00269 | 0.009307 | -0.0452 | -0.01739 | 0.022366 | 0.010282 | 0.003544 | -0.00767 | -0.07668 | 0.022311 |
| 25 | -0.0549 | -0.00467 | -0.02314 | 0.034617 | 0.037782 | 0.040217 | -0.03996 | 0.02321 | -0.00759 | -0.01021 | 0.057703 | 0.007526 | 0.095109 | -0.01139 | -0.02128 | -0.00589 | -0.06043 | 0.004466 | 0.011198 | 0.080725 | -0.06266 | -0.02638 | -0.01634 | 0.048036 | -0.02759 |
| 26 | -0.05473 | -0.0064 | 0.001208 | 0.013379 | 0.013296 | 0.025426 | -0.03232 | 0.01118 | -0.00443 | -0.0178 | 0.043517 | -0.00811 | 0.086453 | 0.032814 | -0.02733 | 0.017829 | 0.009714 | 0.024117 | -0.02246 | 0.064657 | -0.04278 | -0.07264 | -0.03393 | 0.023502 | 0.002422 |
| 27 | -0.03418 | 0.0111 | 0.017514 | -0.03708 | 0.011963 | -0.00496 | -0.01276 | 0.051605 | 0.024899 | -0.01967 | -0.01682 | 0.000331 | 0.015114 | -0.00883 | -0.00524 | -0.0162 | -0.0284 | -0.04618 | 0.021794 | -0.05783 | -0.01792 | 0.01759 | 0.048721 | 0.028904 | 0.038131 |
| 28 | -0.03184 | 0.001692 | 0.012218 | -0.04534 | 0.018782 | -0.00885 | 0.003053 | 0.044826 | 0.028931 | -0.02032 | -0.02708 | 0.010487 | 0.024058 | 0.007875 | -0.02625 | -0.02313 | -0.00186 | -0.02322 | -0.00253 | -0.03607 | 0.00834 | -0.01105 | 0.042154 | 0.036399 | 0.04593 |
| 29 | -0.0247 | -0.0138 | 0.002527 | -0.01267 | 0.000813 | 0.015002 | 0.014155 | -0.01361 | -0.02129 | 0.043361 | 0.004308 | 0.066357 | -0.01328 | -0.0163 | -0.01834 | 0.010607 | 0.00603 | 0.055252 | 0.027 | 0.03245 | -0.01706 | 0.029436 | -0.03037 | 0.002131 | 0.022626 |
| 30 | -0.02965 | -0.01513 | 0.007768 | -0.01821 | 0.005569 | 0.019848 | 0.023564 | -0.01417 | -0.02191 | 0.054316 | -0.01035 | 0.074635 | -0.00747 | -0.00457 | -0.0146 | 0.017484 | 0.019435 | 0.060634 | 0.014697 | 0.031533 | -0.00196 | 0.008558 | -0.03421 | -0.00145 | 0.02928 |
| 31 | -0.0395 | -0.09187 | 0.072761 | 0.015433 | 0.049291 | -0.06108 | -0.02678 | 0.018221 | -0.03215 | -0.06108 | 0.014012 | 0.025133 | 0.101976 | -0.04468 | 0.092221 | -0.04642 | 0.006728 | 0.024504 | 0.010245 | -0.08358 | -0.02832 | 0.017495 | -0.00927 | 0.017146 | -0.10465 |
| 32 | -0.03986 | -0.09317 | 0.084341 | 0.021374 | 0.024315 | -0.08857 | -0.01467 | 0.01291 | -0.04807 | -0.05605 | -0.01073 | 0.033104 | 0.112711 | -0.00521 | 0.077685 | -0.04865 | 0.053624 | 0.019601 | -0.03212 | -0.08858 | 0.009372 | -0.04014 | -0.03625 | -0.00054 | -0.10425 |

*Fig 13: Left matrix*

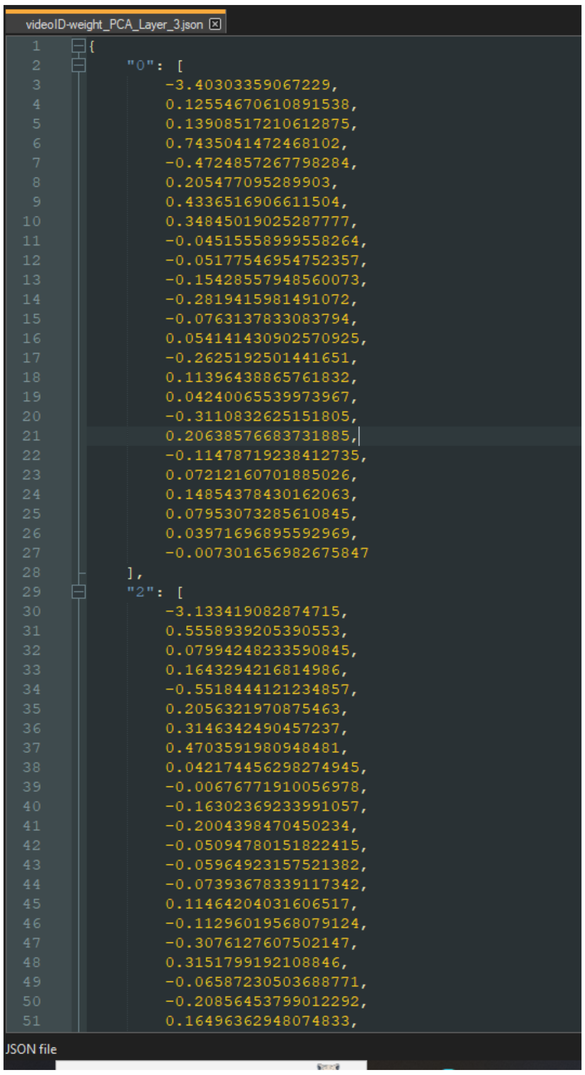
'Fig 13' is the left matrix of the PCA components we have got. The rows are the number of features, and the columns are the latent features.



| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1.251562 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0.098827 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0.079193 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0.067582 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0.046102 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0.041902 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0.037536 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.032696 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.031663 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.026877 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.02291 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.020871 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.018596 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.01767 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.015355 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.015041 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.013977 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.012937 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.011529 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.011466 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.010873 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.009775 | 0 | 0 | 0 |
| 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.009135 | 0 | 0 |
| 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.008747 | 0 |
| 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.00797 |

*Fig 14:Core matrix*

'Fig 14' is the core matrix of the PCA components. We know that the core matrix is diagonal in nature with the size of kxk (k=25 in this case), hence we get a square matrix. Here k is the number of latent semantic features.


videoID-weight_PCA_Layer_3.json

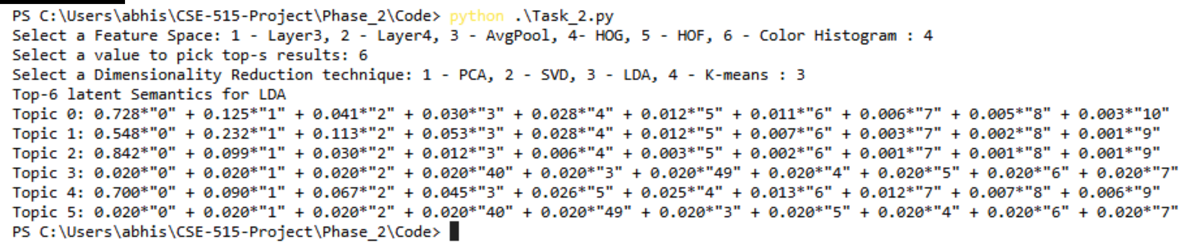
```
{
    "0": [
        -3.40303359067229,
        0.12554670610891538,
        0.13908517210612875,
        0.7435041472468102,
        -0.4724857267798284,
        0.20547709528990З,
        0.4336516906611504,
        0.34845019025287777,
        -0.045155589995558264,
        -0.05177546954752357,
        -0.15428557948560073,
        -0.2819415981491072,
        -0.0763137833083794,
        0.054141430902570925,
        -0.2625192501441651,
        0.11396438865761832,
        0.04240065539973967,
        -0.3110832625151805,
        0.20638576683731885,
        -0.11478719238412735,
        0.07212160701885026,
        0.14854378430162063,
        0.07953073285610845,
        0.03971696895592969,
        -0.007301656982675847
    ],
    "2": [
        -3.133419082874715,
        0.5558939205390553,
        0.07994248233590845,
        0.1643294216814986,
        -0.5518444121234857,
        0.2056321970875463,
        0.3146342490457237,
        0.4703591980948481,
        0.042174456298274945,
        -0.00676771910056978,
        -0.16302369233991057,
        -0.2004398470450234,
        -0.05094780151822415,
        -0.05964923157521382,
        -0.07393678339117342,
        0.11464204031606517,
        -0.11296019568079124,
        -0.3076127607502147,
        0.3151799192108846,
        -0.06587230503688771,
        -0.20856453799012292,
        0.16496362948074833,
```

JSON file

*Fig15:videoID-weight pair*

'Fig 15' shows the VideoID-weight pair. As seen, it is for the even videos. The key is the video ID for each video, and the values are the latent semantic features. Since we have 25 latent features, we are having 25 values in the form of a list.

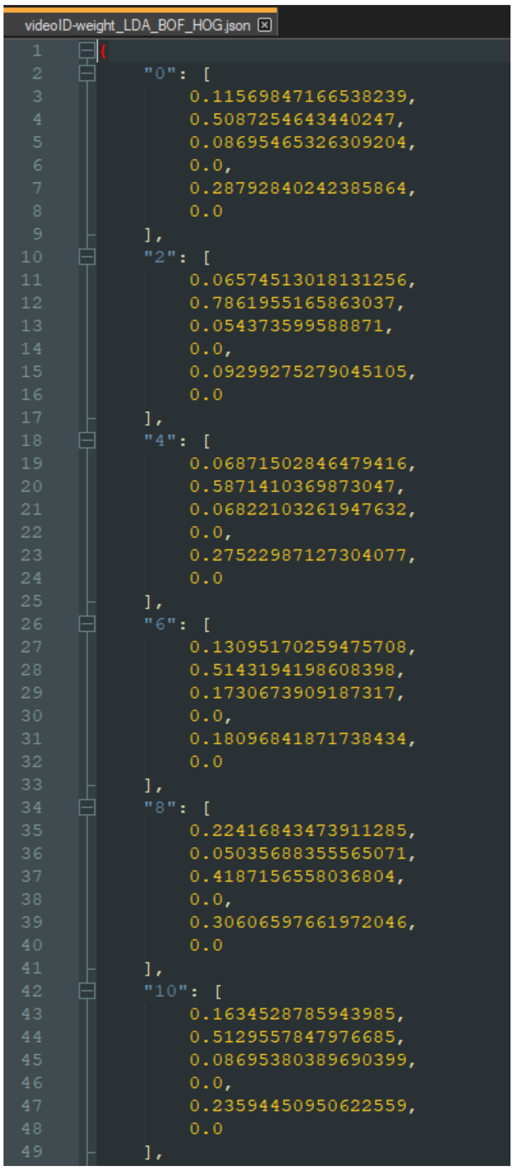
## Input - 2:



```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_2.py
Select a Feature Space: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram : 4
Select a value to pick top-s results: 6
Select a Dimensionality Reduction technique: 1 - PCA, 2 - SVD, 3 - LDA, 4 - K-means : 3
Top-6 latent Semantics for LDA
Topic 0: 0.728*"0" + 0.125*"1" + 0.041*"2" + 0.030*"3" + 0.028*"4" + 0.012*"5" + 0.011*"6" + 0.006*"7" + 0.005*"8" + 0.003*"10"
Topic 1: 0.548*"0" + 0.232*"1" + 0.113*"2" + 0.053*"3" + 0.028*"4" + 0.012*"5" + 0.007*"6" + 0.003*"7" + 0.002*"8" + 0.001*"9"
Topic 2: 0.842*"0" + 0.099*"1" + 0.030*"2" + 0.012*"3" + 0.006*"4" + 0.003*"5" + 0.002*"6" + 0.001*"7" + 0.001*"8" + 0.001*"9"
Topic 3: 0.020*"0" + 0.020*"1" + 0.020*"2" + 0.020*"40" + 0.020*"3" + 0.020*"49" + 0.020*"4" + 0.020*"5" + 0.020*"6" + 0.020*"7"
Topic 4: 0.700*"0" + 0.090*"1" + 0.067*"2" + 0.045*"3" + 0.026*"5" + 0.025*"4" + 0.013*"6" + 0.012*"7" + 0.007*"8" + 0.006*"9"
Topic 5: 0.020*"0" + 0.020*"1" + 0.020*"2" + 0.020*"40" + 0.020*"49" + 0.020*"3" + 0.020*"5" + 0.020*"4" + 0.020*"6" + 0.020*"7"
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> █
```

*Fig 16: LDA using HoG*

'Fig 16' describes the LDA which is performed using HoG feature space. Top 's' results are displayed.

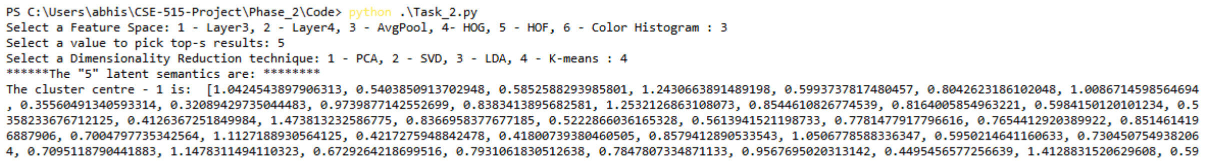


```
{
    "0": [
        0.11569847166538239,
        0.5087254643440247,
        0.08695465326309204,
        0.0,
        0.28792840242385864,
        0.0
    ],
    "2": [
        0.06574513018131256,
        0.7861955165863037,
        0.054373599588871,
        0.0,
        0.09299275279045105,
        0.0
    ],
    "4": [
        0.06871502846479416,
        0.5871410369873047,
        0.06822103261947632,
        0.0,
        0.27522987127304077,
        0.0
    ],
    "6": [
        0.13095170259475708,
        0.5143194198608398,
        0.1730673909187317,
        0.0,
        0.18096841871738434,
        0.0
    ],
    "8": [
        0.22416843473911285,
        0.05035688355565071,
        0.4187156558036804,
        0.0,
        0.30606597661972046,
        0.0
    ],
    "10": [
        0.1634528785943985,
        0.5129557847976685,
        0.08695380389690399,
        0.0,
        0.23594450950622559,
        0.0
    ],
```

*Fig 17: VideoID-Weight Pairs*

'Fig 17' shows the video ID weight pair using the LDA method. Since we are getting this of s=6, each key (even video ID) will be having 6 values(LDA resultant weights).

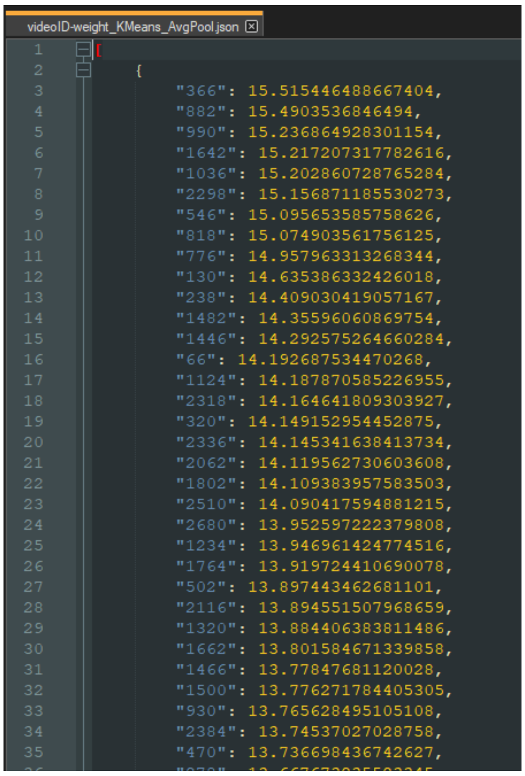
## Input - 3: KMeans



```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_2.py
Select a Feature Space: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram : 3
Select a value to pick top-s results: 5
Select a Dimensionality Reduction technique: 1 - PCA, 2 - SVD, 3 - LDA, 4 - K-means : 4
******The "5" latent semantics are: ********
The cluster centre - 1 is:  [1.0424543897906313, 0.5403850913702948, 0.5852588293985801, 1.2430663891489198, 0.5993737817480457, 0.8042623186102048, 1.0086714598564694
, 0.35560491340593314, 0.32089429735044483, 0.9739877142552699, 0.8383413895682581, 1.2532126863108073, 0.8544610826774539, 0.8164005854963221, 0.5984150120101234, 0.5
358233676712125, 0.4126367251849984, 1.473813232586775, 0.8366958377677185, 0.5222866036165328, 0.5613941521198733, 0.7781477917796616, 0.7654412920389922, 0.851461419
6887906, 0.7004797735342564, 1.1127188930564125, 0.4217275948842478, 0.41800739380460505, 0.8579412890533543, 1.0506778588336347, 0.5950214641160633, 0.730450754938206
4, 0.7095118790441883, 1.1478311494110323, 0.6729264218699516, 0.7931061830512638, 0.7847807334871133, 0.9567695020313142, 0.4495456577256639, 1.4128831520629608, 0.59
```

*Fig 18: K-means using AveragePool*

The above figure(Fig 18) shows the cluster coordinates of the AveragePool layer using k-means clusterization. Here, 5 latent semantics using the clusters are found.



videoID-weight_KMeans_AvgPool.json

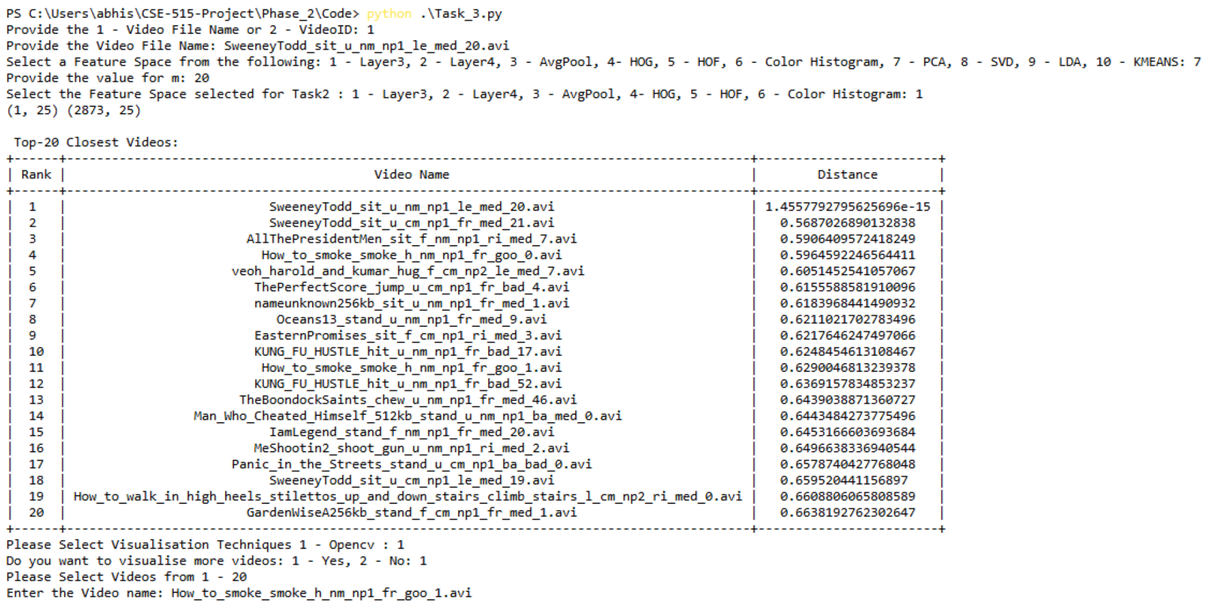
```json
[
    {
        "366": 15.515446488667404,
        "882": 15.4903536846494,
        "990": 15.236864928301154,
        "1642": 15.21720731778261 6,
        "1036": 15.202860728765284,
        "2298": 15.156871185530273,
        "546": 15.095653585758626,
        "818": 15.074903561756125,
        "776": 14.957963313268344,
        "130": 14.635386332426018,
        "238": 14.409030419057167,
        "1482": 14.3559606086975 4,
        "1446": 14.292575264660284,
        "66": 14.192687534470268,
        "1124": 14.187870585226955,
        "2318": 14.164641809303927,
        "320": 14.149152954452875,
        "2336": 14.145341638413734,
        "2062": 14.119562730603608,
        "1802": 14.109383957583503,
        "2510": 14.090417594881215,
        "2680": 13.952597222379808,
        "1234": 13.946961424774516,
        "1764": 13.919724410690078,
        "502": 13.897443462681101,
        "2116": 13.894551507968659,
        "1320": 13.884406383811486,
        "1662": 13.801584671339858,
        "1466": 13.77847681120028,
        "1500": 13.776271784405305,
        "930": 13.765628495105108,
        "2384": 13.74537027028758,
        "470": 13.736698436742627,
```

Fig 19: VideoID-Weight pairs using K-means

Fig 19 shows us the videoID-weight pairs in the form of a dictionary. Here, the key is the video ID and the values are the weights obtained using the K means on AveragePool layer.

## Task-3:

## <u>Input -1:</u>



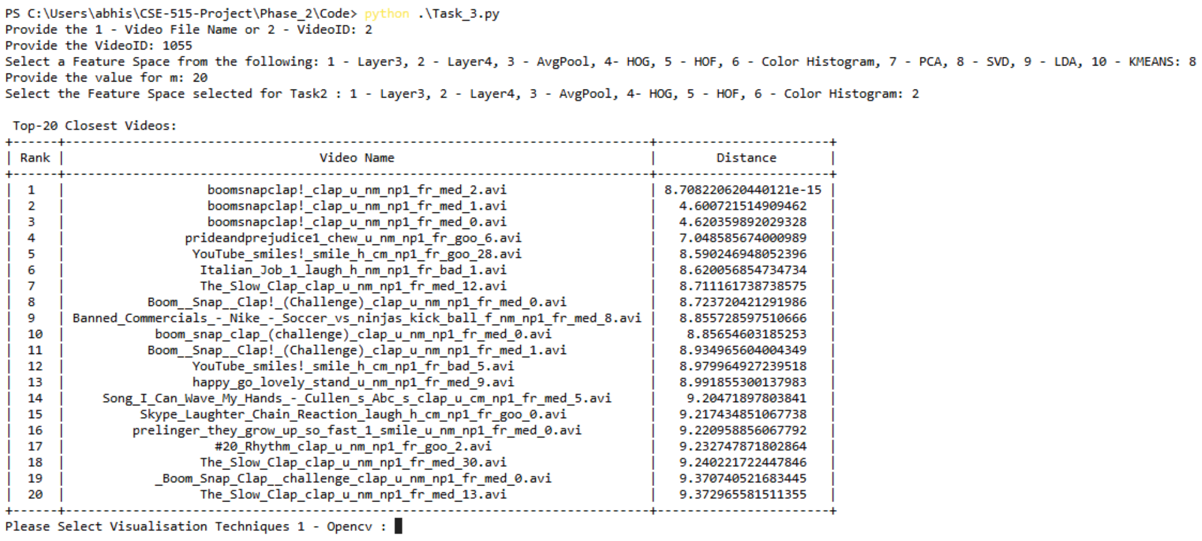
```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_3.py
Provide the 1 - Video File Name or 2 - VideoID: 1
Provide the Video File Name: SweeneyTodd_sit_u_nm_np1_le_med_20.avi
Select a Feature Space from the following: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram, 7 - PCA, 8 - SVD, 9 - LDA, 10 - KMEANS: 7
Provide the value for m: 20
Select the Feature Space selected for Task2 : 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram: 1
(1, 25) (2873, 25)

 Top-20 Closest Videos:
+------+--------------------------------------------------------------------------------------+------------------------+
| Rank |                                      Video Name                                      |        Distance        |
+------+--------------------------------------------------------------------------------------+------------------------+
|  1   |                        SweeneyTodd_sit_u_nm_np1_le_med_20.avi                        | 1.4557792795625696e-15 |
|  2   |                        SweeneyTodd_sit_u_cm_np1_fr_med_21.avi                        |   0.5687026890132838   |
|  3   |                     AllThePresidentMen_sit_f_nm_np1_ri_med_7.avi                     |   0.5906409572418249   |
|  4   |                       How_to_smoke_smoke_h_nm_np1_fr_goo_0.avi                       |   0.5964592246564411   |
|  5   |                   veoh_harold_and_kumar_hug_f_cm_np2_le_med_7.avi                    |   0.6051452541057067   |
|  6   |                      ThePerfectScore_jump_u_cm_np1_fr_bad_4.avi                      |   0.6155588581910096   |
|  7   |                      nameunknown256kb_sit_u_nm_np1_fr_med_1.avi                      |   0.6183968441490932   |
|  8   |                         Oceans13_stand_u_nm_np1_fr_med_9.avi                         |   0.6211021702783496   |
|  9   |                      EasternPromises_sit_f_cm_np1_ri_med_3.avi                       |   0.6217646247497066   |
|  10  |                      KUNG_FU_HUSTLE_hit_u_nm_np1_fr_bad_17.avi                       |   0.6248454613108467   |
|  11  |                       How_to_smoke_smoke_h_nm_np1_fr_goo_1.avi                       |   0.6290046813239378   |
|  12  |                      KUNG_FU_HUSTLE_hit_u_nm_np1_fr_bad_52.avi                       |   0.6369157834853237   |
|  13  |                     TheBoondockSaints_chew_u_nm_np1_fr_med_46.avi                    |   0.6439038871360727   |
|  14  |               Man_Who_Cheated_Himself_512kb_stand_u_nm_np1_ba_med_0.avi              |   0.6443484273775496   |
|  15  |                        IamLegend_stand_f_nm_np1_fr_med_20.avi                        |   0.6453166603693684   |
|  16  |                      MeShootin2_shoot_gun_u_nm_np1_ri_med_2.avi                      |   0.6496638336940544   |
|  17  |                   Panic_in_the_Streets_stand_u_cm_np1_ba_bad_0.avi                   |   0.6578740427768048   |
|  18  |                        SweeneyTodd_sit_u_cm_np1_le_med_19.avi                        |   0.659520441156897    |
|  19  | How_to_walk_in_high_heels_stilettos_up_and_down_stairs_climb_stairs_l_cm_np2_ri_med_0.avi |   0.6608806065808589   |
|  20  |                     GardenWiseA256kb_stand_f_cm_np1_fr_med_1.avi                     |   0.6638192762302647   |
+------+--------------------------------------------------------------------------------------+------------------------+
Please Select Visualisation Techniques 1 - Opencv : 1
Do you want to visualise more videos: 1 - Yes, 2 - No: 1
Please Select Videos from 1 - 20
Enter the Video name: How_to_smoke_smoke_h_nm_np1_fr_goo_1.avi
```

*Fig 20: Closest Videos using PCA*

Fig 20 shows the closest videos that are obtained using PCA latent semantic features over the feature space Layer 3. Here, the value of 'm' is 20 and we have put in the video name directly. The top 20 closest videos are displayed and visualized upon need. The video of the action type sit was provided. By using PCA, it was observed that the first 3 videos correspond to the same action type which is sit.

## <u>**Input -2:**</u>



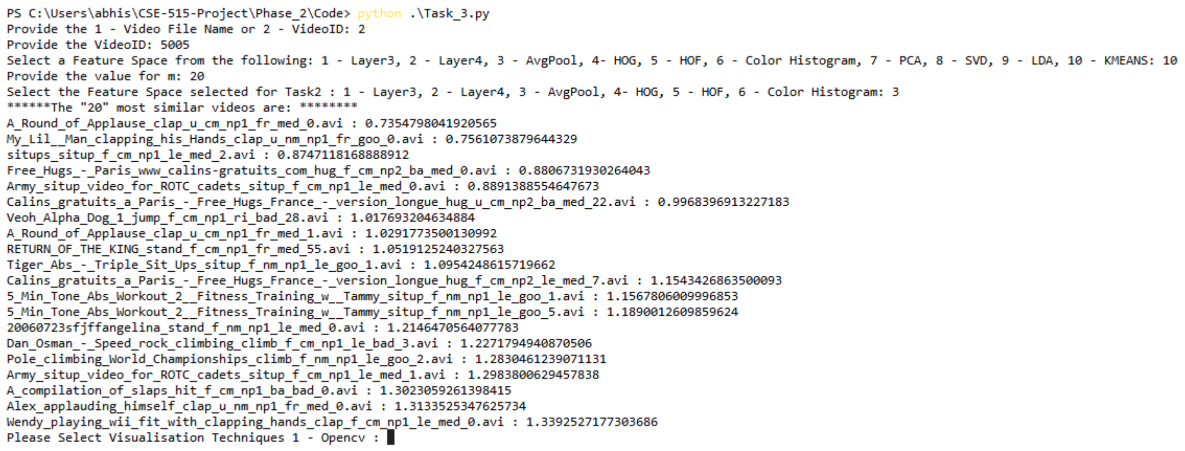
```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_3.py
Provide the 1 - Video File Name or 2 - VideoID: 2
Provide the VideoID: 1055
Select a Feature Space from the following: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram, 7 - PCA, 8 - SVD, 9 - LDA, 10 - KMEANS: 8
Provide the value for m: 20
Select the Feature Space selected for Task2 : 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram: 2

 Top-20 Closest Videos:
+------+-------------------------------------------------------------------------+---------------------+
| Rank |                              Video Name                                 |       Distance      |
+------+-------------------------------------------------------------------------+---------------------+
|  1   |                boomsnapclap!_clap_u_nm_np1_fr_med_2.avi                 | 8.708220620440121e-15 |
|  2   |                boomsnapclap!_clap_u_nm_np1_fr_med_1.avi                 |  4.600721514909462  |
|  3   |                boomsnapclap!_clap_u_nm_np1_fr_med_0.avi                 |  4.620359892029328  |
|  4   |              prideandprejudice1_chew_u_nm_np1_fr_goo_6.avi              |  7.048585674000989  |
|  5   |               YouTube_smiles!_smile_h_cm_np1_fr_goo_28.avi              |  8.590246948052396  |
|  6   |                Italian_Job_1_laugh_h_nm_np1_fr_bad_1.avi                |  8.620056854734734  |
|  7   |                The_Slow_Clap_clap_u_nm_np1_fr_med_12.avi                |  8.711161738738575  |
|  8   |         Boom__Snap__Clap!_(Challenge)_clap_u_nm_np1_fr_med_0.avi        |  8.723720421291986  |
|  9   | Banned_Commercials_-_Nike_-_Soccer_vs_ninjas_kick_ball_f_nm_np1_fr_med_8.avi |  8.855728597510666  |
|  10  |          boom_snap_clap_(challenge)_clap_u_nm_np1_fr_med_0.avi          |   8.85654603185253  |
|  11  |         Boom__Snap__Clap!_(Challenge)_clap_u_nm_np1_fr_med_1.avi        |  8.934965604004349  |
|  12  |               YouTube_smiles!_smile_h_cm_np1_fr_bad_5.avi               |  8.979964927239518  |
|  13  |               happy_go_lovely_stand_u_nm_np1_fr_med_9.avi               |  8.991855300137983  |
|  14  |    Song_I_Can_Wave_My_Hands_-_Cullen_s_Abc_s_clap_u_cm_np1_fr_med_5.avi |   9.20471897803841  |
|  15  |         Skype_Laughter_Chain_Reaction_laugh_h_cm_np1_fr_goo_0.avi       |  9.217434851067738  |
|  16  |       prelinger_they_grow_up_so_fast_1_smile_u_nm_np1_fr_med_0.avi      |  9.220958856067792  |
|  17  |                  #20_Rhythm_clap_u_nm_np1_fr_goo_2.avi                  |  9.232747871802864  |
|  18  |                The_Slow_Clap_clap_u_nm_np1_fr_med_30.avi                |  9.240221722447846  |
|  19  |          _Boom_Snap_Clap__challenge_clap_u_nm_np1_fr_med_0.avi          |  9.370740521683445  |
|  20  |                The_Slow_Clap_clap_u_nm_np1_fr_med_13.avi                |  9.372965581511355  |
+------+-------------------------------------------------------------------------+---------------------+
Please Select Visualisation Techniques 1 - Opencv : █
```

*Fig 21: Similar Videos using SVD*

As shown in Fig 21, the latent semantic feature selected was SVD and the feature space that was considered was Layer4, using this the top 20 similar videos are visualized. The given video is of type clap and the top 20 videos that are listed have the videos that are of the same action type. The first three videos are of action type clap which specifies that dimensionality reduction using SVD captured accurate results.

## <u>**Input-3:**</u>



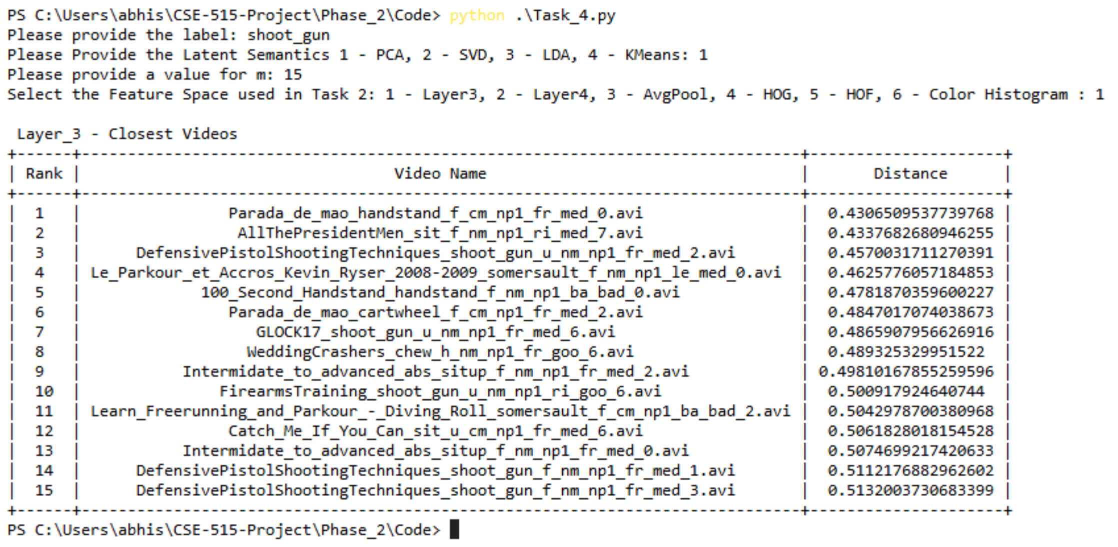
```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_3.py
Provide the 1 - Video File Name or 2 - VideoID: 2
Provide the VideoID: 5005
Select a Feature Space from the following: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram, 7 - PCA, 8 - SVD, 9 - LDA, 10 - KMEANS: 10
Provide the value for m: 20
Select the Feature Space selected for Task2 : 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4- HOG, 5 - HOF, 6 - Color Histogram: 3
******The "20" most similar videos are: ********
A_Round_of_Applause_clap_u_cm_np1_fr_med_0.avi : 0.7354798041920565
My_Lil__Man_clapping_his_Hands_clap_u_nm_np1_fr_goo_0.avi : 0.7561073879644329
situps_situp_f_cm_np1_le_med_2.avi : 0.8747118168888912
Free_Hugs_-_Paris_www_calins-gratuits_com_hug_f_cm_np2_ba_med_0.avi : 0.8806731930264043
Army_situp_video_for_ROTC_cadets_situp_f_cm_np1_le_med_0.avi : 0.8891388554647673
Calins_gratuits_a_Paris_-_Free_Hugs_France_-_version_longue_hug_u_cm_np2_ba_med_22.avi : 0.9968396913227183
Veoh_Alpha_Dog_1_jump_f_cm_np1_ri_bad_28.avi : 1.017693204634884
A_Round_of_Applause_clap_u_cm_np1_fr_med_1.avi : 1.0291773500130992
RETURN_OF_THE_KING_stand_f_cm_np1_fr_med_55.avi : 1.0519125240327563
Tiger_Abs_-_Triple_Sit_Ups_situp_f_nm_np1_le_goo_1.avi : 1.0954248615719662
Calins_gratuits_a_Paris_-_Free_Hugs_France_-_version_longue_hug_f_cm_np2_le_med_7.avi : 1.1543426863500093
5_Min_Tone_Abs_Workout_2__Fitness_Training_w__Tammy_situp_f_nm_np1_le_goo_1.avi : 1.1567806009996853
5_Min_Tone_Abs_Workout_2__Fitness_Training_w__Tammy_situp_f_nm_np1_le_goo_5.avi : 1.1890012609859624
20060723sfjffangelina_stand_f_nm_np1_le_med_0.avi : 1.2146470564077783
Dan_Osman_-_Speed_rock_climbing_climb_f_cm_np1_le_bad_3.avi : 1.2271794940870506
Pole_climbing_World_Championships_climb_f_nm_np1_le_goo_2.avi : 1.2830461239071131
Army_situp_video_for_ROTC_cadets_situp_f_cm_np1_le_med_1.avi : 1.2983800629457838
A_compilation_of_slaps_hit_f_cm_np1_ba_bad_0.avi : 1.3023059261398415
Alex_applauding_himself_clap_u_nm_np1_fr_med_0.avi : 1.3133525347625734
Wendy_playing_wii_fit_with_clapping_hands_clap_f_cm_np1_le_med_0.avi : 1.3392527177303686
Please Select Visualisation Techniques 1 - Opencv : █
```

*Fig 22: Similar videos using K-means*

As shown in the above figure(Fig 22), we are getting the most similar videos using the K-means as a latent semantic feature over the feature space of AveragePool. Here, we get the ‘m’ most similar videos (here m= 20). The given video file is of nontarget videos of action-type push ups. Since the similarity was found from ‘m’ similar target videos the videos relevant to a specific action were not found.

## Task-4:

### Input-1:



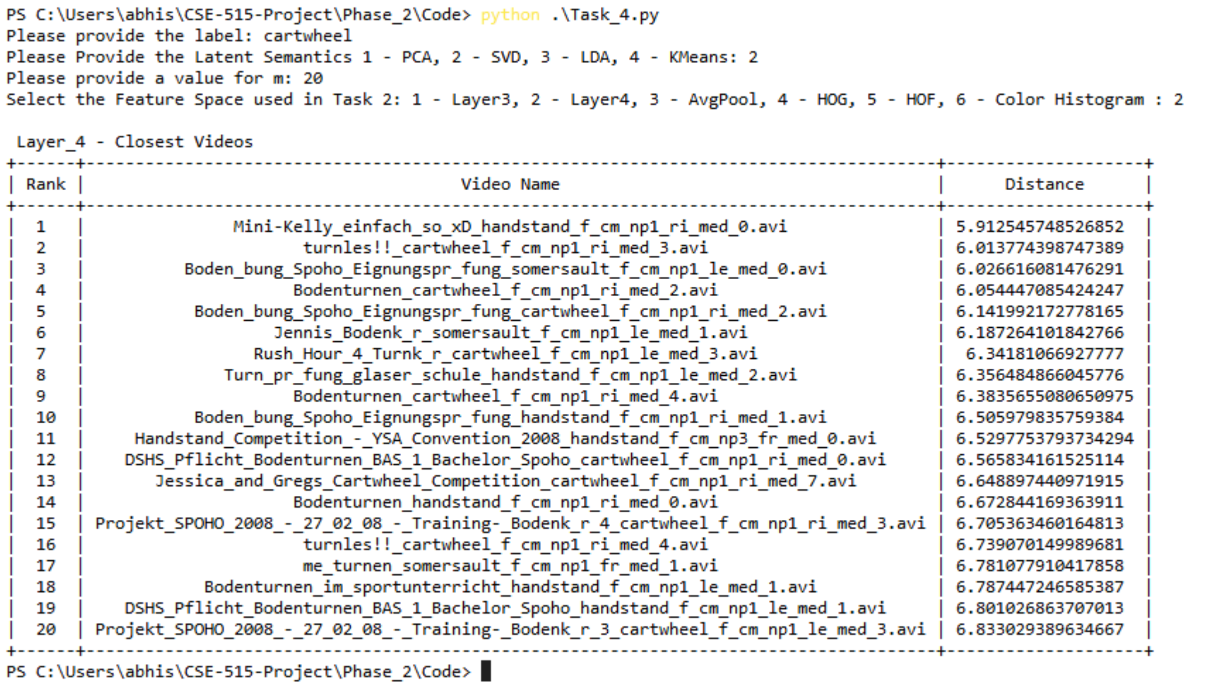
```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_4.py
Please provide the label: shoot_gun
Please Provide the Latent Semantics 1 - PCA, 2 - SVD, 3 - LDA, 4 - KMeans: 1
Please provide a value for m: 15
Select the Feature Space used in Task 2: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4 - HOG, 5 - HOF, 6 - Color Histogram : 1

 Layer_3 - Closest Videos
+------+--------------------------------------------------------------------------+--------------------+
| Rank |                                Video Name                                |      Distance      |
+------+--------------------------------------------------------------------------+--------------------+
|  1   |               Parada_de_mao_handstand_f_cm_np1_fr_med_0.avi               | 0.4306509537739768 |
|  2   |               AllThePresidentMen_sit_f_nm_np1_ri_med_7.avi                | 0.4337682680946255 |
|  3   |       DefensivePistolShootingTechniques_shoot_gun_u_nm_np1_fr_med_2.avi    | 0.4570031711270391 |
|  4   | Le_Parkour_et_Accros_Kevin_Ryser_2008-2009_somersault_f_nm_np1_le_med_0.avi | 0.4625776057184853 |
|  5   |            100_Second_Handstand_handstand_f_nm_np1_ba_bad_0.avi            | 0.4781870359600227 |
|  6   |               Parada_de_mao_cartwheel_f_cm_np1_fr_med_2.avi               | 0.4847017074038673 |
|  7   |                 GLOCK17_shoot_gun_u_nm_np1_fr_med_6.avi                   | 0.4865907956626916 |
|  8   |                WeddingCrashers_chew_h_nm_np1_fr_goo_6.avi                 | 0.489325329951522  |
|  9   |          Intermidate_to_advanced_abs_situp_f_nm_np1_fr_med_2.avi          |0.49810167855259596 |
|  10  |              FirearmsTraining_shoot_gun_u_nm_np1_ri_goo_6.avi              | 0.500917924640744  |
|  11  | Learn_Freerunning_and_Parkour_-_Diving_Roll_somersault_f_cm_np1_ba_bad_2.avi | 0.5042978700380968 |
|  12  |               Catch_Me_If_You_Can_sit_u_cm_np1_fr_med_6.avi               | 0.5061828018154528 |
|  13  |          Intermidate_to_advanced_abs_situp_f_nm_np1_fr_med_0.avi          | 0.5074699217420633 |
|  14  |       DefensivePistolShootingTechniques_shoot_gun_f_nm_np1_fr_med_1.avi    | 0.5112176882962602 |
|  15  |       DefensivePistolShootingTechniques_shoot_gun_f_nm_np1_fr_med_3.avi    | 0.5132003730683399 |
+------+--------------------------------------------------------------------------+--------------------+
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> █
```

*Fig 23: Closest Videos for given Label, Feature Space, Latent Semantics(PCA)*

Fig 23 shows how for a user-defined label, Feature Space, and Latent Semantics we can get the closest videos. For the above case, the label chosen is shoot_gun, using PCA, over the Layer 3 feature space. For the given action label, we could find videos of the same action type, however, the 1st action type is handstand but the 3rd action type is shoot_gun. The similar videos consist of other action types like cartwheel, situp, sit, and somersault.

### Input-2:



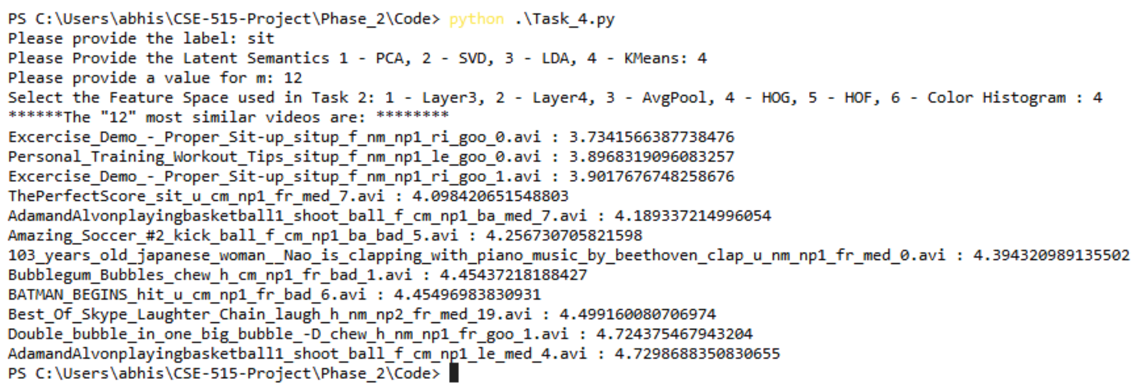
```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_4.py
Please provide the label: cartwheel
Please Provide the Latent Semantics 1 - PCA, 2 - SVD, 3 - LDA, 4 - KMeans: 2
Please provide a value for m: 20
Select the Feature Space used in Task 2: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4 - HOG, 5 - HOF, 6 - Color Histogram : 2

 Layer_4 - Closest Videos
+------+------------------------------------------------------------------------------+-------------------+
| Rank |                                  Video Name                                  |     Distance      |
+------+------------------------------------------------------------------------------+-------------------+
|  1   |              Mini-Kelly_einfach_so_xD_handstand_f_cm_np1_ri_med_0.avi              | 5.912545748526852 |
|  2   |                    turnles!!_cartwheel_f_cm_np1_ri_med_3.avi                     | 6.013774398747389 |
|  3   |         Boden_bung_Spoho_Eignungspr_fung_somersault_f_cm_np1_le_med_0.avi         | 6.026616081476291 |
|  4   |                   Bodenturnen_cartwheel_f_cm_np1_ri_med_2.avi                    | 6.054447085424247 |
|  5   |         Boden_bung_Spoho_Eignungspr_fung_cartwheel_f_cm_np1_ri_med_2.avi          | 6.141992172778165 |
|  6   |                Jennis_Bodenk_r_somersault_f_cm_np1_le_med_1.avi                 | 6.187264101842766 |
|  7   |               Rush_Hour_4_Turnk_r_cartwheel_f_cm_np1_le_med_3.avi                |  6.34181066927777 |
|  8   |            Turn_pr_fung_glaser_schule_handstand_f_cm_np1_le_med_2.avi             | 6.356484866045776 |
|  9   |                   Bodenturnen_cartwheel_f_cm_np1_ri_med_4.avi                    | 6.3835655080650975 |
|  10  |          Boden_bung_Spoho_Eignungspr_fung_handstand_f_cm_np1_ri_med_1.avi          | 6.505979835759384 |
|  11  |      Handstand_Competition_-_YSA_Convention_2008_handstand_f_cm_np3_fr_med_0.avi      | 6.5297753793734294 |
|  12  |     DSHS_Pflicht_Bodenturnen_BAS_1_Bachelor_Spoho_cartwheel_f_cm_np1_ri_med_0.avi     | 6.565834161525114 |
|  13  |       Jessica_and_Gregs_Cartwheel_Competition_cartwheel_f_cm_np1_ri_med_7.avi        | 6.648897440971915 |
|  14  |                   Bodenturnen_handstand_f_cm_np1_ri_med_0.avi                    | 6.672844169363911 |
|  15  | Projekt_SPOHO_2008_-_27_02_08_-_Training-_Bodenk_r_4_cartwheel_f_cm_np1_ri_med_3.avi | 6.705363460164813 |
|  16  |                    turnles!!_cartwheel_f_cm_np1_ri_med_4.avi                     | 6.739070149989681 |
|  17  |                   me_turnen_somersault_f_cm_np1_fr_med_1.avi                    | 6.781077910417858 |
|  18  |           Bodenturnen_im_sportunterricht_handstand_f_cm_np1_le_med_1.avi           | 6.787447246585387 |
|  19  |     DSHS_Pflicht_Bodenturnen_BAS_1_Bachelor_Spoho_handstand_f_cm_np1_le_med_1.avi     | 6.801026863707013 |
|  20  | Projekt_SPOHO_2008_-_27_02_08_-_Training-_Bodenk_r_3_cartwheel_f_cm_np1_le_med_3.avi | 6.833029389634667 |
+------+------------------------------------------------------------------------------+-------------------+
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> █
```

*Fig 24:Closest Videos for given Label, Feature Space, Latent Semantics(SVD)*

Fig 24 shows how for a user-defined label, Feature Space, and Latent Semantics we can get the closest videos. For the above case, the label chosen is cartwheel, using SVD, over the Layer 4 feature space. The given video label is cartwheel, and similar videos are found for the same action label.

**<u>Input-3:</u>**



```
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code> python .\Task_4.py
Please provide the label: sit
Please Provide the Latent Semantics 1 - PCA, 2 - SVD, 3 - LDA, 4 - KMeans: 4
Please provide a value for m: 12
Select the Feature Space used in Task 2: 1 - Layer3, 2 - Layer4, 3 - AvgPool, 4 - HOG, 5 - HOF, 6 - Color Histogram : 4
******The "12" most similar videos are: ********
Excercise_Demo_-_Proper_Sit-up_situp_f_nm_np1_ri_goo_0.avi : 3.7341566387738476
Personal_Training_Workout_Tips_situp_f_nm_np1_le_goo_0.avi : 3.8968319096083257
Excercise_Demo_-_Proper_Sit-up_situp_f_nm_np1_ri_goo_1.avi : 3.9017676748258676
ThePerfectScore_sit_u_cm_np1_fr_med_7.avi : 4.098420651548803
AdamandAlvonplayingbasketball1_shoot_ball_f_cm_np1_ba_med_7.avi : 4.189337214996054
Amazing_Soccer_#2_kick_ball_f_cm_np1_ba_bad_5.avi : 4.256730705821598
103_years_old_japanese_woman__Nao_is_clapping_with_piano_music_by_beethoven_clap_u_nm_np1_fr_med_0.avi : 4.394320989135502
Bubblegum_Bubbles_chew_h_cm_np1_fr_bad_1.avi : 4.45437218188427
BATMAN_BEGINS_hit_u_cm_np1_fr_bad_6.avi : 4.45496983830931
Best_Of_Skype_Laughter_Chain_laugh_h_nm_np2_fr_med_19.avi : 4.499160080706974
Double_bubble_in_one_big_bubble_-D_chew_h_nm_np1_fr_goo_1.avi : 4.724375467943204
AdamandAlvonplayingbasketball1_shoot_ball_f_cm_np1_le_med_4.avi : 4.7298688350830655
PS C:\Users\abhis\CSE-515-Project\Phase_2\Code>
```

*Fig 25- Lists the similar videos for a given class label and a feature matrix.*

As a sample input, the video label that was provided was sit and the latent semantics provided was KMeans. For the given feature space as HoG, listed the top 12 similar videos, where the videos sit, laugh, shoot_ball, and situp. From the 5 videos, are a mix of sit up and sit, as the actions are similar to each other. However, some of the actions like laugh, and shoot_ball were not expected for action sit.

## 10. Conclusions:

As a part of Phase-2 as a learning step, the feature models that are extracted as part of Phase-1 are revisited and corrected to extract the accurate feature. During this phase, we have worked on various tasks that include the prediction of target video labels for a given video. Implemented Dimensionality Reduction techniques like PCA, SVD, LDA, and KMeans. Using the features that are extracted as a part of the dimensionality reduction, the nearest videos for a given video have been computed using distance metrics like Euclidean, Manhattan, and cosine similarity. It was observed that the actions that are too close to each other are part of the similar videos list.

## Appendix:

1. Abhishek Mulasi: Task-0, Task-2, Task-3, Task-4, Readme, Documentation.
2. Sri Rupin Potula: Task-0, Task-1, Task-2, Task-3. Task-4, Readme, Documentation.
3. Vibhu Dixit: Documentation.
4. Vibha Swaminathan: -

## Bibliography:

[1] https://www.sqlite.org/whentouse.html
[2] https://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/
[3] https://www.ezyzip.com/open-extract-rar-file-online.html